

nomos Dokumentation

Release 1.1.73

29.05.2015

Willkommen in der Onlinedokumentation der nomos IoT Engine

Die nomos IoT Software stellt als Hintergrundprozess einen TCP-/UDP-Server-Dienst (Daemon) zur Steuerung von Geräten per Netzwerk zur Verfügung. Die Software basiert auf C Code für Linux Systeme. Ein speziell für diese Anwendung definiertes Protokoll ermöglicht es, über eine einfache Netzwerkanbindung per LAN oder WLAN vielfältige Aktionen an diesen Systemen und deren angebundenen Geräte auszuführen und umfangreich zu steuern. Das nomos Protokoll dient hier auch der Vereinheitlichung der Befehlssätze aller angebundenen Geräte. nomos system wird auch unter verschiedenen OEM Namen vertrieben.

Bitte beachten Sie, dass die Abbildungen teilweise nicht den aktuellen Softwarestand aufzeigen und daher leicht abweichen können.

KAPITEL 1

Einleitung

Lernen Sie die universelle nomos system IoT Software-Engine kennen. Die nomos IoT Software steuert alle Geräte unabhängig von Kommunikationsprotokollen, Standards oder proprietärer Software.

Als "Schweizer Taschenmesser" der Automation ist die nomos IoT Software-Engine in der Lage sowohl Software als auch Hardware auf eine Kommunikationsebene zu heben und schafft dadurch ein nahtloses Automationssystem, dass Daten von jeder Platform erfasst, steuert und zustellt.

Die nomos IoT Software-Engine als Antwort auf den rasant wachsenden Automationsmarkt

Die nomos Software wurde so entwickelt, dass sie in der Lage ist neue Standards in der Automation extrem schnell zu adaptieren und das bei einer Größe von nur 613KB.

Anders als die üblichen "Point-to-Point" Automationssysteme wurde nomos entwickelt alle Software, Web-Services, Gebäudeautomation und Multimedia-Komponenten auf nur eine Kommunikationsebene zu heben und liefert dabei den vollen Umfang einer zentralen Steuerung, einschließlich Logiken, Scripts und Events.

In einem OEM bzw. White Label Szenario kann die nomos IoT Software- Engine auf praktisch allen Plattformen eingesetzt werden, einschließlich Low-Power Geräten, oder sogar eingebettet in einem Chipsatz.

Ein Elektronikhersteller hat mit nomos die einzigartige Möglichkeit echte Interoperabilität innerhalb der eigenen Produktgruppen zu schaffen und kann zudem gleichzeitig Anwendungen von Drittanbietern ansteuern.

Die nomos IoT Software-Engine bietet das Potenzial für zahllose Anwendungsbeispiele in der realen Welt und ist in der Lage sich an individuelles Verbraucherverhalten azupassen und dieses zu erlernen.

In einer typischen Morgenroutine erkennt die nomos Software die persönlichen Weck-Vorlieben des Benutzers durch Wearable-Technologie. nomos aktiviert die entsprechenden Makros, schaltet die Phillips HUE Lampen auf die "Energize" Szene und spielt die Sonos Playlist " Morning Classic" bei 37% Lautstärke. Die Nespresso Maschine brüht den ersten Kaffee und der Fernseher schaltet automatisch stumm auf die Frühstücksnachrichten.

Das System erkennt automatisch wenn der letzte Bewohner das Haus verlassen hat und schaltet alle Geräte aus. Im Auto auf dem Weg zur Arbeit übernimmt nomos die Spotify Playlisten, zeigt den Status der Wohnung in einer Echtzeit 3-D Kartenumgebung und weist den Weg zur nächstgelegenen E-Tankstelle. Nach einem langen Arbeitstag schickt nomos beim Verlassen des Büros eine Push Nachricht mit der kalkulierten Ankunftszeit an Familie oder Freunde, oder gibt bei Bedarf Auskunft über die Parkplatzsituation am Zielort. Das beschriebene Szenario ist nur eines von unzähligen Einsatzmöglichkeiten der nomos IoT Engine, weitere Use Cases finden Sie unter http://www.nomos-system.com.

Über nomos system

Gegründet von Veteranen in der Hausautomation im Jahr 2010 ist nomos system das führende Unternehmen für innovative Automationslösungen, mit derzeit mehr als 2.000 lizenzierten Projekten und 500 aktiven Systemintegratoren im nomos Forum.

Unter Verwendung der eigenen Software-Engine, hat nomos system zahlreiche Projekte im privaten wie auch kommerziellen Bereich in Europa, den Vereinigten Staaten und Asien realisiert. Ausgerichtet für jegliche Anwendung, ob privat oder kommerziell in der Automobilindustrie, Luftfahrt oder auf einer Yacht, beliefert nomos OEM Partner in der Automation auf der ganzen Welt. Weitere Informationen finden Sie unter http://www.nomos-system.com.

KAPITEL 2

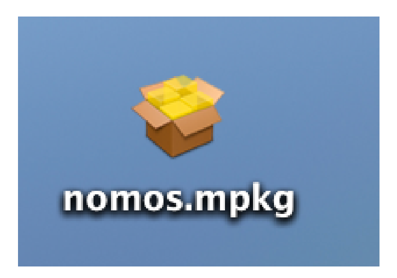
Grundkonfiguration, Lizenzierung und Steuerung

Für die Grundkonfiguration (Netzwerk Konfiguration) des nomos systems stehen entsprechende Eingabemasken zur Verfügung. Bei der Verwendung von Mac Systemen finden Sie diese Eingabemaske nach der Softwareinstallation in der Systemsteuerung. Bei Verwendung der Linux Versionen erreichen Sie die Eingabemasken via Webpanel. Beachten Sie hierbei, dass die Linux Systeme erst lizensiert werden müssen, bevor der Webserver eine Konfiguration zulässt. Die Mac Version bietet einige zus. Konfigurationsmöglichkeiten wie zB. auch die Einstellbarkeit des Projektpfades.

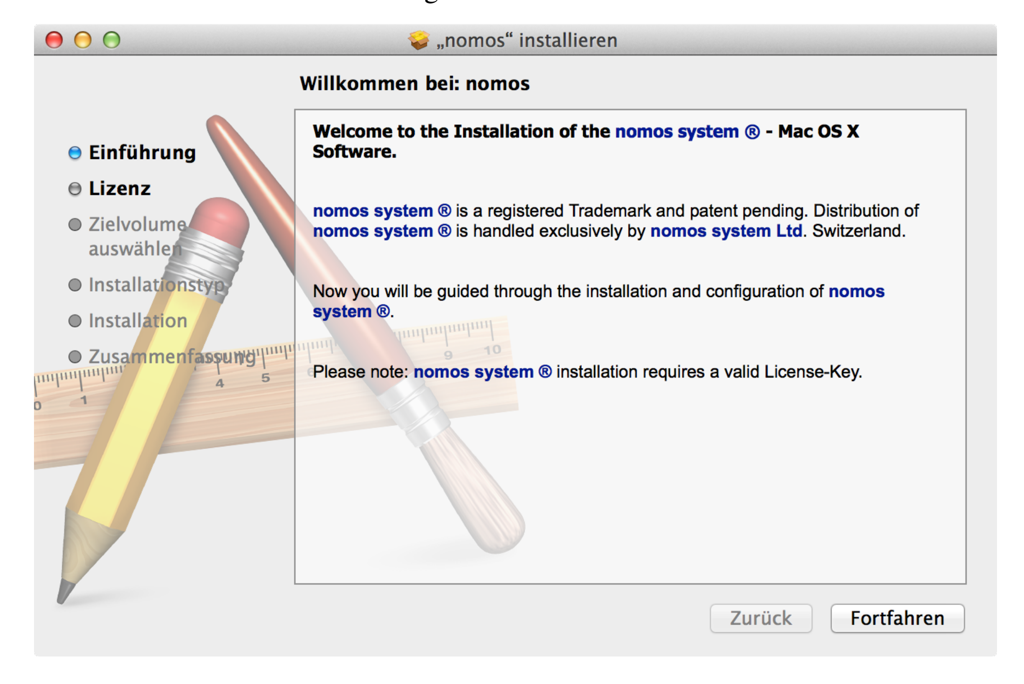
2.1 Grundkonfiguration

2.1.1 Installation Mac OSX

Die nomos Linux Versionen werden vorinstalliert auf entsprechende Hardware Plattformen geliefert. Die Apple OS X Version muss von Ihnen installiert werden. Sie starten die Apple OSX Installation von nomos durch Doppelklick auf das nomos Paket:



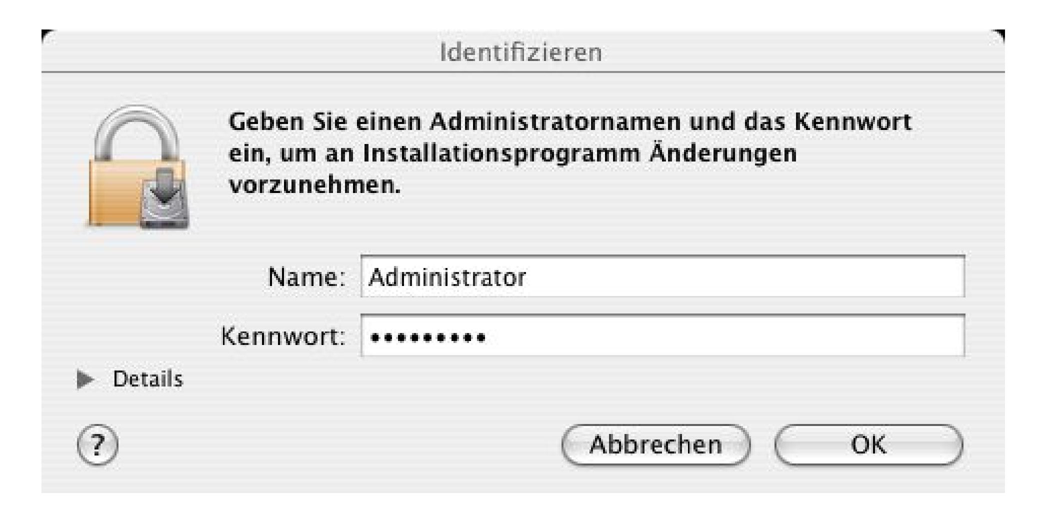
Sie werden daraufhin durch die Installation geführt:



Klicken Sie auf „Fortfahren".

Lesen Sie die erscheinende Lizenzvereinbarung aufmerksam durch und klicken Sie erneut auf „Fortfahren". Bestätigen Sie die nachfolgende Abfrage, wenn Sie der Lizenz zustimmen, durch einen Klick auf „Agree". Das Installationsprogramm zeigt Ihnen im Anschluss daran an, dass nomos auf der Systempartition installiert wird. Eine Installation auf einer anderen Partition ist systembedingt nicht möglich. Bestätigen Sie dies durch einen Klick auf „Fortfahren". In dem nun erscheinenden Fenster klicken sie bitte auf „Aktualisieren", um die Installation durchzuführen.

Zur Installation von nomos sind Administrator Rechte erforderlich. Bitte geben Sie die erforderlichen Daten ein:



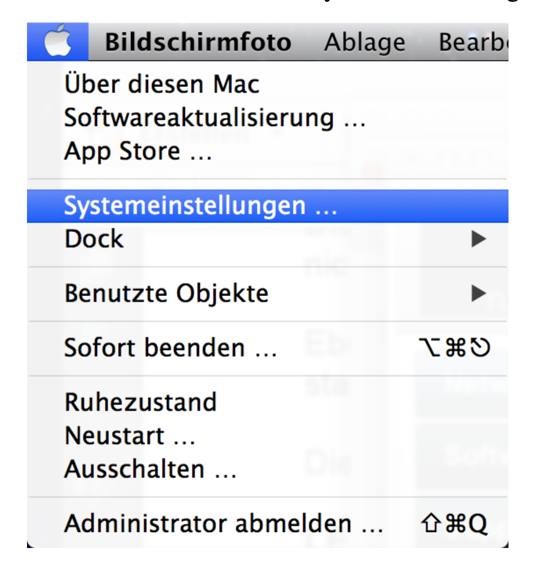
Bestätigen Sie Ihre Eingaben mit „OK“. nomos wird nun installiert. Bestätigen Sie den Abschluss der Installation durch einen Klick auf „Schließen“.

Herzlichen Glückwunsch! nomos ist nun auf Ihrem System installiert. Bitte beachten Sie, dass die Installation von nomos immer benutzerbezogen stattfindet.

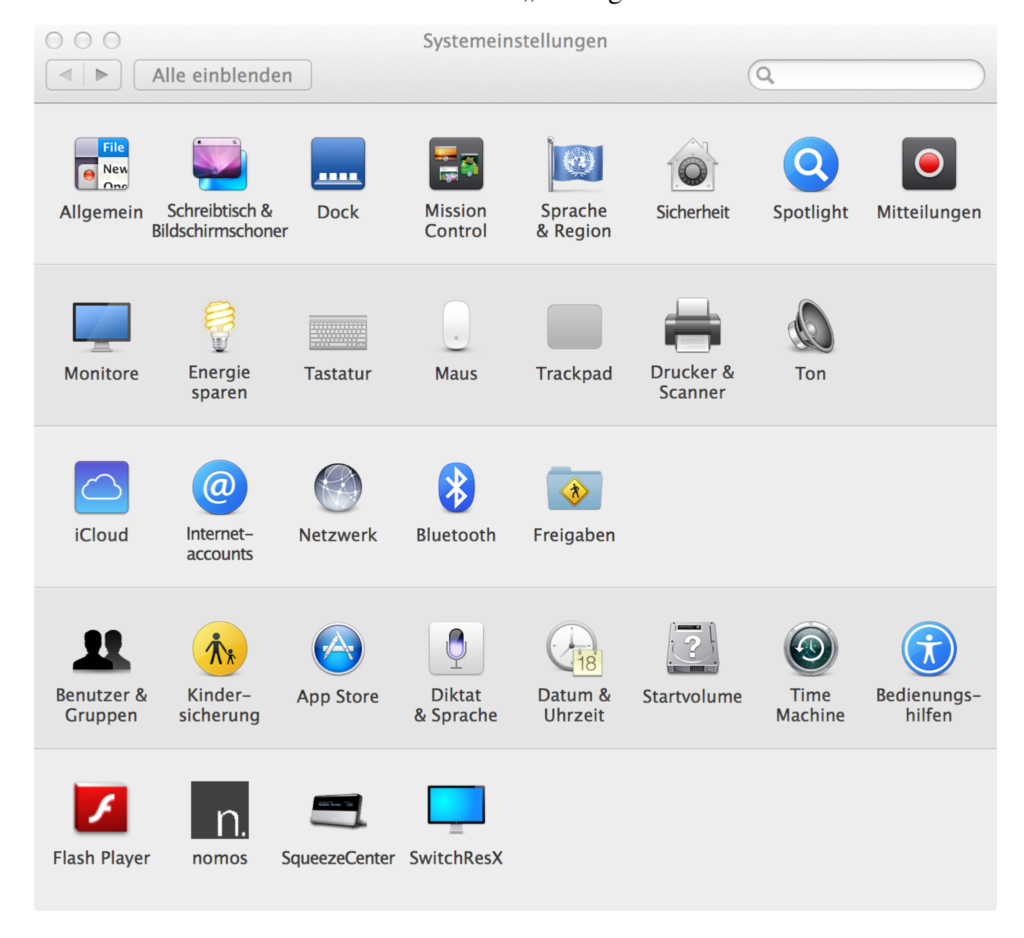
Eine Mehrfachinstallation von nomos unter verschiedenen Benutzern auf einem System ist nicht zu empfehlen, da dies u.U. zu Funktionsstörungen führen kann.

2.1.2 Konfiguration Mac

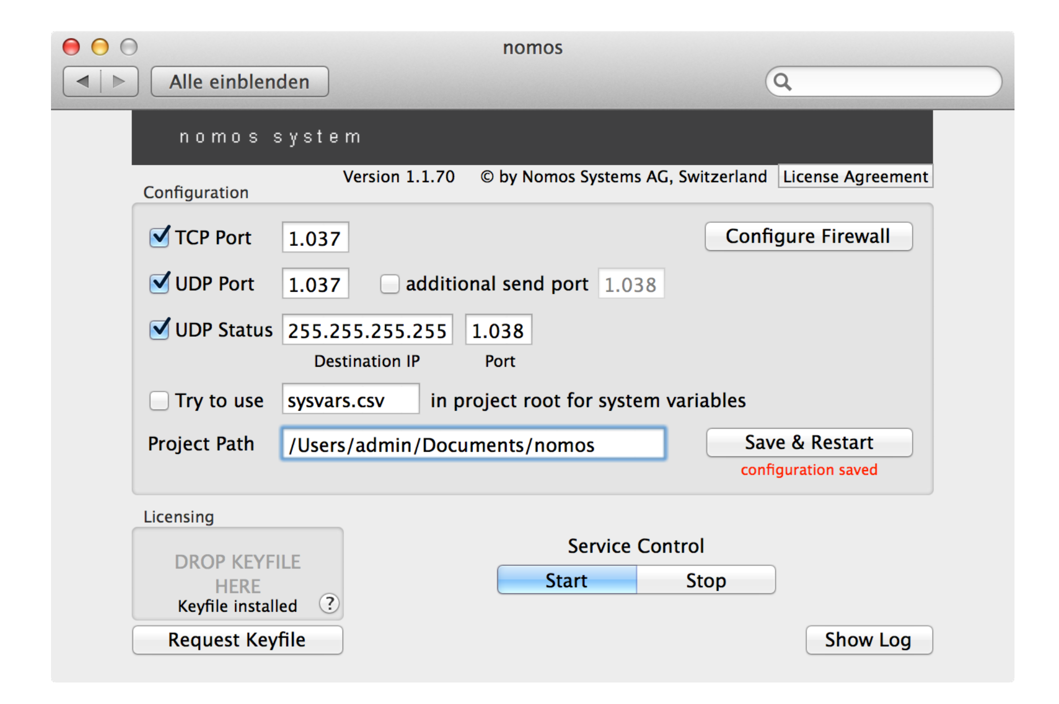
Zur Konfiguration von nomos Mac öffnen Sie bitte die Systemeinstellungen.



Im nun erscheinenden Fenster klicken Sie bitte unter„Sonstige“ im unteren Bereich auf das nomos Icon:



Sie befinden sich nun in der Konfiguration (Preference Pane) von nomos:



Im Feld „Configuration“ werden alle zum Betrieb des nomos erforderlichen Netzwerkeinstellungen vorgenommen. Änderungen an der Konfiguration werden durch einen Klick auf „Save & Restart“ dauerhaft abgespeichert und der Dienst ggfs. neu gestartet, so dass Änderungen an der Konfiguration sofort übernommen werden. nomos kann sowohl TCP- als auch UDP-Verbindungen annehmen. Die einzelnen Dienste werden durch Setzen eines Hakens aktiviert.

TCP Port Gibt den Port an, auf dem nomos auf eingehende TCP-Verbindungen wartet Gültiger Wertebereich: `1024-1038` und `1040-65535`. Eine bestehende Verbindung auf einem TCP Port hat sperrende Wirkung, so dass während dessen keine UDP-Kommandos verarbeitet werden.

UDP Port Gibt den Port an, auf dem nomos auf UDP-Pakete wartet.Gültiger Wertebereich: `1024-65535`

additional send port Durch Aktivieren dieser Option aktivieren Sie das zusätzliche Senden derAntwortsequenz von nomos an den Client auf einem zusätzlichen Port. Diese Option ist als Kompatibilitätsmodus für Clients mit eingeschränkter Netzwerk- funktionalität zu verstehen und sollte nur im Bedarfsfall aktiviert werden. Der zusätzliche Send-Port entspricht immer dem gewählten UDP Port+1.

UDP Status Geben Sie hier die IP (Broadcastadressen sind erlaubt) und den Port an, an den Statusinformationen über das System gesendet werden. nomos sendet die Statusinformationen automatisch bei Änderungen, ohne dass ein Client diese Informationen explizit abfragen muss.

Try to use... Hier kann der Standard Zugriff auf die `sysvars.csv` verändert werden. Wird diese Option nicht genutzt, liegt die `sysvars.csv` unter `.\misc` unterhalb des Projektverzeichnis (default). Bei Verwenden dieser Option liegt die Datei direkt im root des Projektverzeichnis. Darüber hinaus kann ebenfalls der Dateiname angepasst werden.
Diese Option wird bei komplexeren Projekten verwendet sofern man Systemvariablen für die dynamische Konfiguration verwendet.

Der TCP Port `1039` ist für Debug- und Logging-Funktionen fest reserviert und deshalb nicht auswählbar. Um diese Funktion zu nutzen, bauen Sie bitte eine zusätzliche TCP-Verbindung zu nomos auf diesem Port auf. nomos wird Ihnen daraufhin Informationen über die Abarbeitung von empfangenen Kommandos senden. Es können bis zu 16 Clients gleichzeitig auf den Port `1039` zugreifen.

2.1.3 Konfiguration Linux (nomos Box/Pi)

Einrichtung der nomos Box Pi Version per Wizard
Einrichtung der nomos Box Pi Version per Konsole (Terminal)

Funktional sind die nomos Box Versionen 1:1 vergleichbar mit den Funktionen auf dem MAC. Ausgenommen sind natürlich die MAC eigenen Klassen zur Steuerung der internen Applikationen, wie `ITUNES`, `TV`, `WINDOW` usw. Ebenso sind einige Einschränkungen in der SYS Klasse wie zB `SETVOL` oder `SAY` oder `ENABLESS` zu berücksichtigen.

Die nomos Box/Pi Systeme sind rein als Gateway konzipiert bzw. als Server für eben HTML5 oder `mremote` (CommandFusion). Selbstverständlich wird auch der CommandFusion GUI Upload sowie der `remote` Service für AppleTV und die „entfernte" iTunes Anbindung wie auch UPNP und XPL unterstützt.

Der nomos Box/Pi HTML5 Service transcodiert hierbei automatisch die GUI Designer Projektdatei auf HTML5. Somit steht das grafische User Interface jedem HTML5 fähigen Webrowser zur Verfügung.

Es stehen zur Zeit zwei unterschiedliche Hardware Varianten zur Verfügung:

nomos Box

Hardware: alix3d2 (PC Engines), 1 LAN / 2 miniPCI / LX800 / 256 MB / USB

Spezifikation:

CPU: 500 MHz AMD Geode LX800

DRAM: 256 MB DDR DRAM

Storage: CompactFlash socket

Power: DC jack or passive POE, min. 7V to max. 20V Three LEDs

Expansion: 2 miniPCI slots, LPC bus

Connectivity: 1 Ethernet channel (Via VT6105M 10/100)

I/O: DB9 serial port, dual USB

Board size: 100 x 160 mm - same as WRAP.2E

Firmware: tinyBIOS

Die Box verfügt an der Front über drei Status LED's:

LED 1 - (rechts neben der Netzversorgung) zeigt den Boot Vorgang an: LED EIN: System bootet
LED blinkt: System gestartet (an der Blinkfrequenz erkennt man die Systemauslastung Daemon Heartbeat)

LED 2 - zeigt an, dass der NOMOS Service gestartet ist: LED AUS: Kein Service gestartet (darf eigentlich nie der Fall sein)
LED EIN: Service gestartet und betriebsbereit.

LED 3 - zeigt den Zugriff auf externen Speicher (zB USB Stick) an: LED AUS: kein Zugrifff
LED EIN: Lese-/Schreibzugriff

Die nomos verfügt über eine interne 4GB Industrial NAND Flash Karte. Die Flash Karte kann nicht einfach getauscht werden, da die Hardware ID der Karte für den Lizenz Schlüssel benötigt wird.

Die nomos Box verfügt neben dem Netzanschluss über einen Ethernet Anschluss, eine RS232 Schnittstelle und auf der Rückseite über zwei USB Anschlüsse. Ebenso unterstützt die nomos Box PoE jedoch nicht den üblichen Standard, der zB bei den gängigen PoE Switches verwendet wird. Um PoE nutzen zu können, wird ein optionaler PoE Injector benötigt.

nomos Pi

Hardware: Raspberry Pi Model B

Spezifikation:

CPU: Broadcom BCM2835 SoC full HD multimedia applications processor

DRAM: 512 MB RAM, 400 MHz

Storage: SD Memory Card Slot (SDHC), kompatibel zu Class 4 und Class 6 Karten

Power: 750mA up to 1.2A @ 5V, 5 Status LEDs

Board size: 85,60mm x 53,98mm x 17mm

Die nomos Pi Box verfügt an der Front über fünf Status LED‘s:

LED 1 - (von oben) Lese-/Schreibzugriffe SD-Karte. Daemon Heartbeat LED blinkt: System gestartet. An der gleichmässigen Blinkfrequenz erkennt man die Systemauslastung. Unregelmässiges Blinken zeigt die Lese-Schreibzugriffe auf die SD-Karte.

LED 2 - (von oben) zeigt den Power Status an: LED EIN: Power.

LED 3, 4, 5 - zeigt den Netzwerkzugrif an LED 3: FDx
LED 4: Link
LED 5:10/100MBit

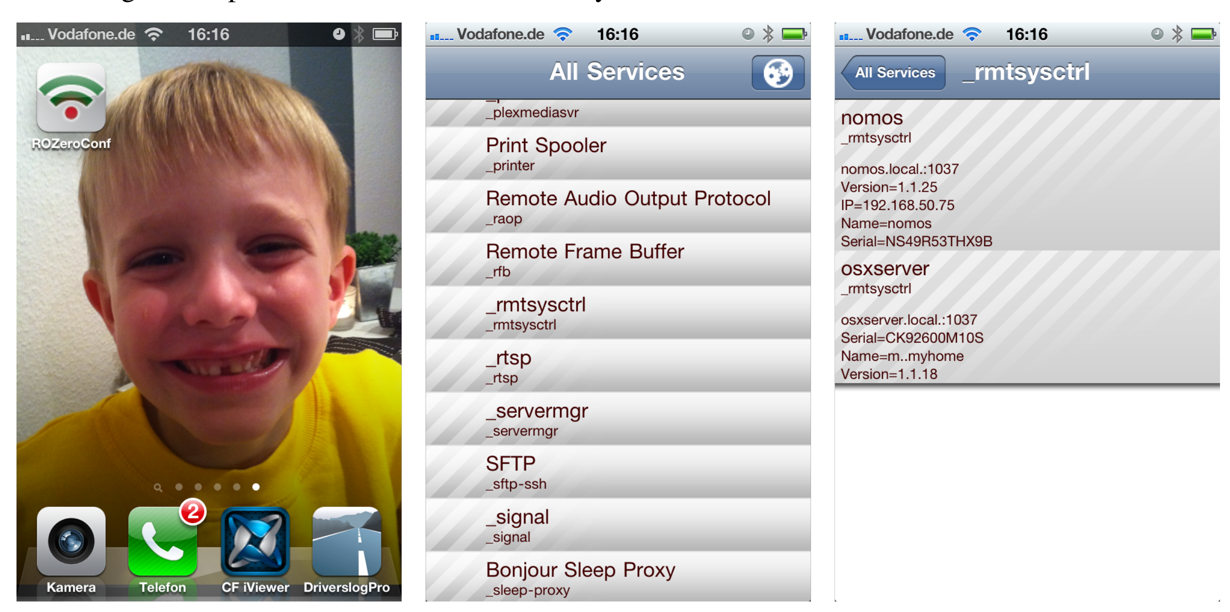
Einrichtung der nomos Box/Pi Version per Wizard

Um die nomos Box/Pi einrichten zu können, müssen Sie zuerst die IP Adresse ermitteln. Ist die IP Adresse bekannt und die nomos Box/Pi ist entsprechend lizenziert, kann die weitere Konfiguration über das R&D Webinterface erfolgen. Es besteht ebenfalls die Möglichkeit, das System manuell einzurichten, zu konfigurieren und zu lizenzieren. Für die manuelle Einrichtung ist grundsätzlich ein FTP Client hilfreich. Wir empfehlen hier z.B. Filezilla. Der sichere Umgang mit dem FTP Client wird vorausgesetzt. Die manuelle Konfiguration wird im weiteren Verlauf noch genauer erläutert.

Die nomos Box (Pi ausgenommen) ist mit einer internen Backup/Restore Funktion ausgestattet. Sofern das OS immer noch startet, kann der Auslieferungszustand, Lizenz ausgenommen, wiederhergestellt werden. Auch diese Möglichkeit wird im weiterem Verlauf noch genauer erörtert.

Für die erstmalige Inbetriebnahme der Box wird ein DHCP-Server im Netzwerk benötigt. Die vergebene IP erfährt man entweder über das Log des DHCP-Servers oder über das ZeroConf-Protokoll (passende iPhone-App: "ZeroConf Browser" von RemObjects Software).

Die Box gibt sich per ZeroConf als Dienst "_rmtsysctrl" zu erkennen:

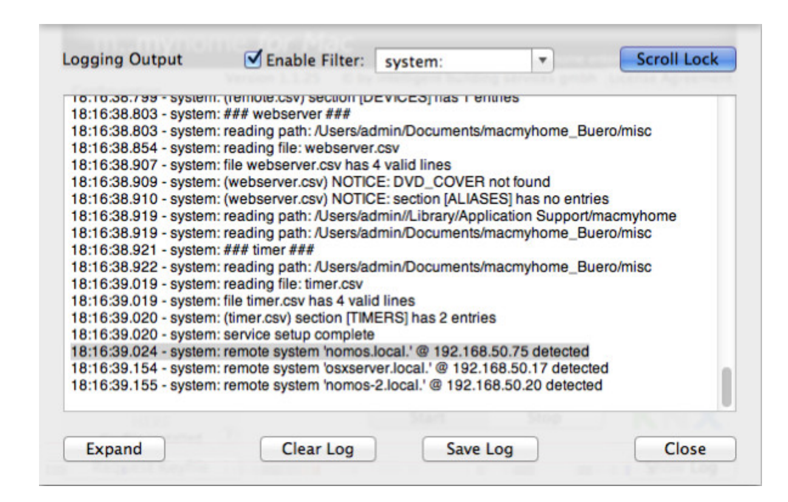


Über den ZeroConf Service wird neben der IP Adresse auch der Kommunikationsport, der Servicename, die Version sowie die Seriennummer ausgegeben.

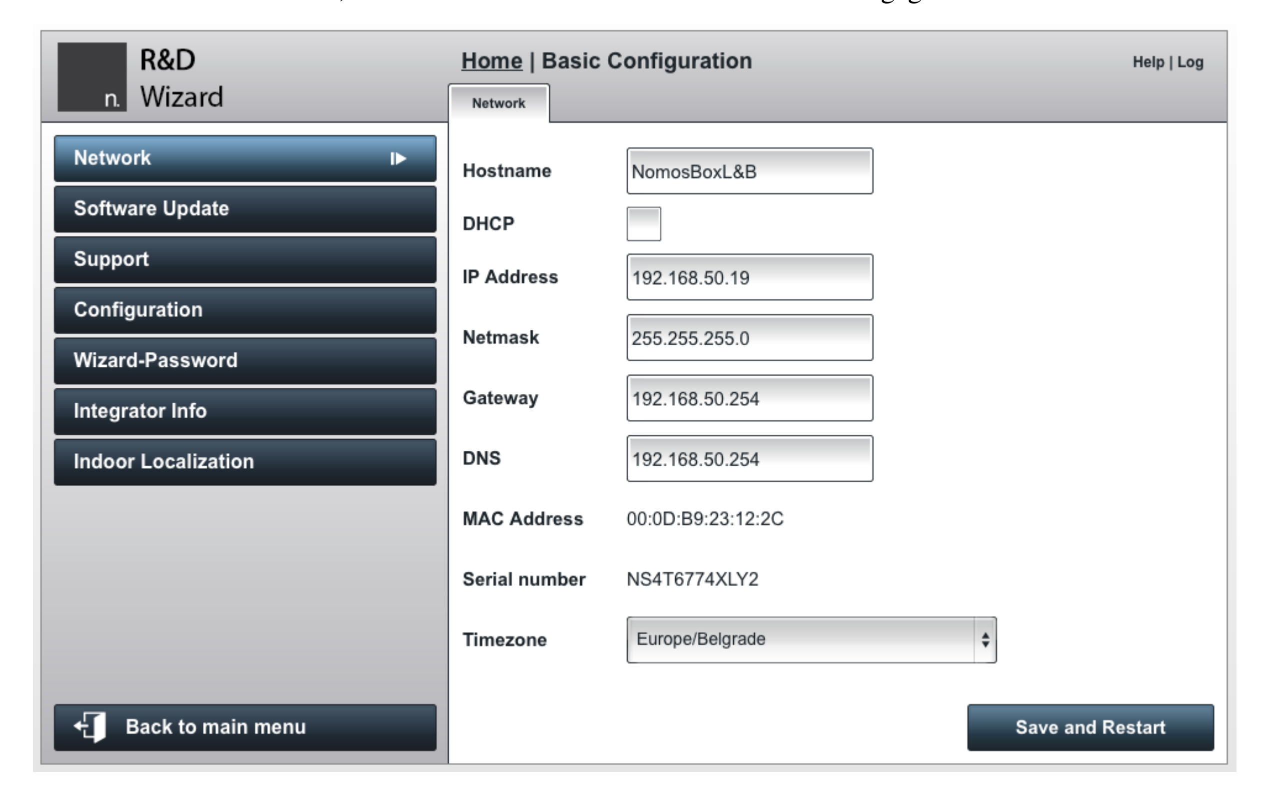
Alternativ erkennt ein laufendes nomos system natürlich ebenfalls die Box. Hierbei das LOG aus der nomos Konsole beobachten, wenn die BOX erfolgreich gestartet ist. Hilfreich ist das Einschalten des Filters. Ggfl. einfach einmal den nomos Service neu starten:

Es erscheint im LOG zB folgende Meldung:

```
18:16:39.024 - system: remote system 'nomos.local.' @ 192.168.50.75 detected
```



Ist die IP Adresse bekannt, kann auf das Webinterface der nomos Box zugegriffen werden:



Auf der „Basis Configuration" Seite des R&D Wizard können nun die individuellen Netzwerkeinstellungen vorgenommen werden. Die Service Ports (`1037-1039`) sind hierbei fest vergeben und können entgegen der Mac OSX Version an dieser Stelle nicht verändert werden. Gleiches gilt für die Verwendung des Projektnamens (nomos).

Wünschen Sie dennoch eine Änderung dieser Standard Einstellungen, müssen Sie eine manuelle Änderung der Systemdatei `nomos.conf` vornehmen.

Neben der Änderung/Anpassung der Netzwerkeinstellungen bietet Ihnen der R&D Wizard auch den Zugriff auf weitere Dienste wie dem Zugriff auf den Software Update Server, der Support Schnittstelle (VPN), eine Eingabemaske zur Sicherung/Wiederherstellung der Konfigurationsdaten, Passwortvergabe für den Zugriff auf den Wizard u.v.m.

Die jeweiligen Anwendungen sind selbsterklärend und werden hier nicht weiter erläutert.

Einrichtung der nomos Box/Pi Version per Konsole (Terminal)

Man kann natürlich auch über das Terminal auf die Box zugreifen und Änderungen vornehmen. Dies sollte jedoch nur dann gemacht werden, wenn man sich mit der Shell auch entsprechend auskennt. Ebenso kann über das Terminal auf das LOG der Box zugegriffen werden bzw. direkte Befehle abgesetzt werden.

Für die manuelle Konfiguration stellt die Box zwei Standard-Dienste bereit: SSH und FTP. Der Login bzw. die Zugangsdaten für beide Dienste lautet:

Login:	root
Passwort:	nomos

Die Zugangsdaten können zZ noch nicht geändert werden.

Die `nomos.conf` für die manuelle Einstellung der Systemports und dem Projektnamen befindet sich im Verzeichnis `/root/nomos/config` und ist wie folgt aufgebaut:

-	TCP_PORT=1037
-	UDP_PORT=1037
-	UDP_PORT_ADD=NO
-	STATUS_PORT=1038
-	STATUS_IP=255.255.255.255
-	PROJECT=nomos

Die Terminal Verbindung mit der Box wird wie folgt hergestellt:

ssh root@[IP Adresse der Box] Terminal öffnen und ssh Verbindung zur Box aufbauen. Die Password Aufforderung mit `nomos` bestätigen.

Nach der Anmeldung gelangt man direkt in das User Verzeichnis der Box `/root`. Nicht zu verwechseln mit dem `root` - Verzeichnis, welches eine Ebene höher liegt.

Vergabe der IP Adresse:

Möchte man der Box dauerhaft eine feste IP vergeben, muss die Datei `/etc/network/interfaces` angepasst werden:

Den Eintrag: `iface eth0 inet dhcp`

mit einem vorangestellten "#" auskommentieren oder löschen

Die IP-Informationen wie folgt eintragen, z.B.:

-	iface eth0 inet static
-	address 192.168.1.128
-	netmask 255.255.255.0
-	broadcast 192.168.1.255
-	gateway 192.168.1.1

Ausserdem sollte man noch die Datei `/etc/resolv.conf` anpassen und einen gültigen Namenserver (DNS-Server) eintragen, z.B:

```
nameserver 192.168.1.1
```

Für die feste Vergabe der IP Adresse, wie auch das Rücksetzen auf DHCP ist ebenfalls ein Shell Script verfügbar, welches einfach per USB Stick eingespielt werden kann. Hierzu finden Sie aktuelle Scripts in unserem support Forum.

2.2 Lizenzierung

Einspielen einer Lizenzdatei Mac

Einspielen einer Lizenzdatei nomos Box Pi

Um nomos in vollem Umfang nutzen zu können, benötigen Sie eine gerätebezogene Lizenzdatei. Ohne Lizenz wird nomos nur im Demo-Modus ausgeführt. Im Demo-Modus ist die volle Funktionalität inklusive einem CommandServer und einem Apple Remote Server verfügbar. Darüber hinaus wird alle 30 - 50 Minuten (zufallsabhängig) die Ausführung gestoppt und muss manuell neu gestartet werden. Der Start von nomos ist in der Demo Version um 30 Sekunden verzögert. Währenddessen erscheint eine entsprechende Einblendung. Die Demo Version ist ausschließlich für die Apple OSX Versionen verfügbar. nomos ist modular aufgebaut und kann beliebig erweitert werden. Das Programmpaket besteht aus einer Basislizenz und den entsprechenden, optionalen Erweiterungen. Zurzeit stehen folgende Module und Systemschnittstellen zur Verfügung:

Basislizenzen Mac:

nomos Gate Grundversion, nicht erweiterbar, für Anwender die einen Apple Computer z.B als einfachen Audiozuspieler oder Medien Client/Set-Top Box verwenden wollen. Beinhaltet einen CommandServer, einen Remote Server, WebServer, einfache Buttonleisten (Standard GUI). Diese Version ist auch zur Steuerung eines einfachen Medicenters (Frontrow/XBMC, eyeTV, ITunes) geeignet.

nomos Gate plus Grundversion, wie vor, jedoch erweiterbar und beinhaltet 10 CommandServer und 3 Remote Server Lizenzen.

Paketlizenzen nomos Box/Pi:

Für die nomos Box Versionen stehen zur Zeit 2 Pakete (nomos Box - Base und nomos Box - Pro Lizenz) zur Verfügung:

nomos Box Base Beinhaltet 3 CommandServer, 1 Apple Remote Server (ATV/ATV2/Homesharing), mRemote Server, HTML5 Server, xPL AddOn, EventServer AddOn, KNX & HS AddOn, UPnP AddOn.

nomos Box Pro Wie nomos Box Base jedoch 10 CommandServer, 5 Apple Remote Server (ATV/ATV2/Homesharing).

Erweiterungen/AddOn's (nur in Verbindung mit der Gate plus Lizenz):

CommandServer AddOn Erweiterung der CommandServer Schnittstelle um jeweils 5 Protokolle (Es sind max. 25 Protokolle je System möglich).

EventServer AddOn Globale Schnittstelle zum Empfangen und Auswerten diverser TCP/UDP Protokolle. (TCP/UDP Listener)

Remote Server AddOn Implementierung des Apple Remote Protokoll zum protokollbasierten Steuern von iTunes (Mac/Windows), AppleTV. Das AddOn ist je Remote Device zu lizensieren. Max. sind 25 Remote Device Verbindungen möglich.

XPL AddOn Einbindung des xPL Protokolls für zB die Ansteuerung der Logitech Squeeze Box.

Extended Buttons AddOn Erweiterte Buttonleisten Steuerung. (max. 25 Buttonleisten (GUI Interface) definierbar. Erweiterter Befehlssatz für die Einrichtung der Buttons. Nur für Mac Systeme relevant.

BAOS/KNX/HS AddOn Building Management Modul für die Einbindung des Weinzierl KNX IP 770 Interface(Objektserver), und/oder GIRA Homeserver KO-Gateway, und/oder KNXnet/IP (Routing oder Tunneling), und/oder die entsprechenden Funktprotokolle Z-Wave und ZigBee. Die Steuerung der Phillips HUE fällt ebenso unter dieser Lizenz.

Streaming AddOn Modul für den eyeTV Streaming Service. Streamt den aktuellen Transponder (4 Kanäle) ins Netzwerk und kann z.B. über VLC empfangen werden.

UPnP AddOn Modul für die Aktivierung des UPnP Protokolls. Dieses AddOn wird zB auch benötigt, wenn Sie ein SONOS System anbinden wollen.

Das Streaming AddOn befindet sich in der Testphase und wird kostenlos und ohne Anspruch auf Vollständigkeit ausgeliefert.

m..Remote Server AddOn Kommunikationsserver für die Einbindung der CommandFusion Schnittstelle. Dieses Modul wird je System, auf dem per mobile Device wie zB iPhone / iPod Touch / iPad zugegriffen werden soll, empfohlen.

Client Lizenzen:

m..Remote Client AddOn Endgeräte Lizenz für die Verwendung der iViewer Client Software von CommandFusion. Je iPhone/iPod Touch/iPad wird eine Lizenz benötigt (nur in Verbindung mit dem m..Remote Server verwendbar)

Wir garantieren einen kostenlosen Update Service von mindestens einem Jahr.

Für OEM Kunden oder Abnehmer im Rahmen einer außerordentlichen Lizenzvereinbarung, können beliebige Lizenz Modelle bzw. Lizenz Pakete generiert werden.

2.2.1 Einspielen einer Lizenzdatei Mac

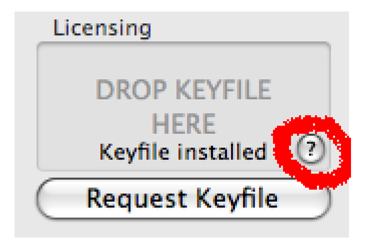
Sollten Sie über keine gültige Lizenzdatei verfügen, können Sie nomos veranlassen, eine entsprechende E- Mail zu generieren. Hierfür muss Ihr auf dem System installierter Email-Client korrekt konfiguriert sein Klicken Sie auf „Request Keyfile". nomos erzeugt daraufhin eine Email mit den zur Lizenzierung notwendigen Systeminformationen und zeigt die erzeugte Email in ihrem Email-Client an. Sollten Sie weitere Anmerkungen zu Ihrer Anfrage haben, können Sie diese selbstverständlich hinzufügen und die E-mail im Anschluss daran absenden. Sie erhalten umgehend Rückmeldung durch unseren Support.

Grundsätzlich wird zur Erzeugung einer Lizenz die Seriennummer des entsprechenden Systems benötigt. Die Seriennummer finden Sie ebenfalls auf dem Gehäuse oder in den Systeminformationen.

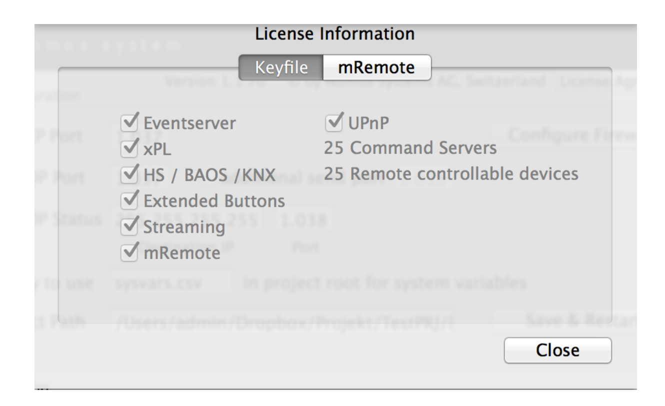
Sollten Sie bereits über eine Lizenz für nomos verfügen, klicken Sie bitte die Lizenzdatei an und ziehen Sie sie bei gedrückter Maustaste auf das Feld „DROP KEYFILE HERE". Dieses Verfahren gilt ebenfalls für die m..Remote Client Lizenzen.



Lassen Sie die Maustaste los. nomos wird Ihnen nun mitteilen, ob es sich um eine gültige Lizenzdatei handelt und die Datei auf Ihren Wunsch installieren.



Innerhalb des Feldes für das Keyfile befindet sich ein kleiner Button mit einem Fragezeichen. Wenn dieser Button angeklickt wird, erscheint ein weiteres Fenster, welches Auskunft über die aktuell installierte Lizenz gibt.



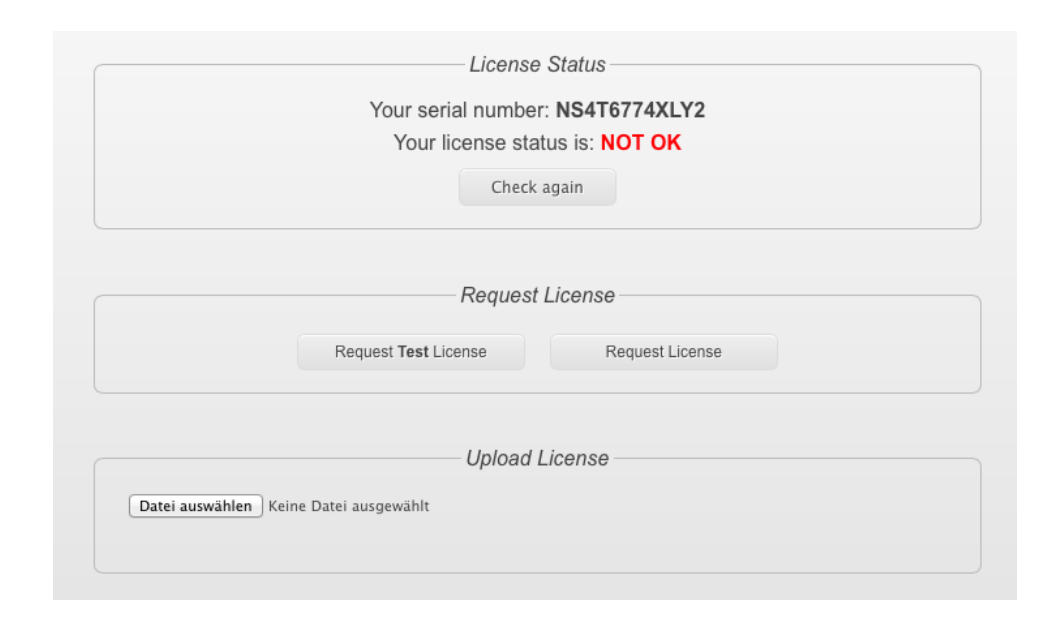
Ist die Lizenz zeitlich limitiert, wird dieses in diesem Fenster ebenfalls angezeigt. In dieser Ansicht finden Sie im Reiter mRemote auch die entsprechend lizenzierten m..Remote Client Lizenzen.

2.2.2 Einspielen einer Lizenzdatei nomos Box/Pi

Die Lizenzdatei für die nomos Box/Pi Versionen können über den Wizard, sofern aktiv, eingespielt werden. Sollte der Wizard noch nicht aktiv sein, kann die Lizenzdatei ebenfalls per FTP übertragen werden. Um eine Lizenzdatei anfordern zu können, benötigen Sie die Seriennummer der nomos Box/Pi. Diese finden Sie u.a. in der Datei `serial.txt` in dem Verzeichnis `/config`.

Haben Sie ein Keyfile erhalten muss die Datei unter dem Namen `nomos.key` in das Verzeichnis `/config` abgelegt werden. Dies kann, wie vor beschrieben, mittels FTP erfolgen. Ist der R&D Wizard aktiv, steht ein Dialog für die Lizenzierung zur Verfügung. Dieser Dialog erscheint automatisch, wenn kein Lizenzfile vorhanden ist. Ebenfalls erhält die Dialogseite alle notwendigen Informationen zur Registrierung der nomos Box/Pi.

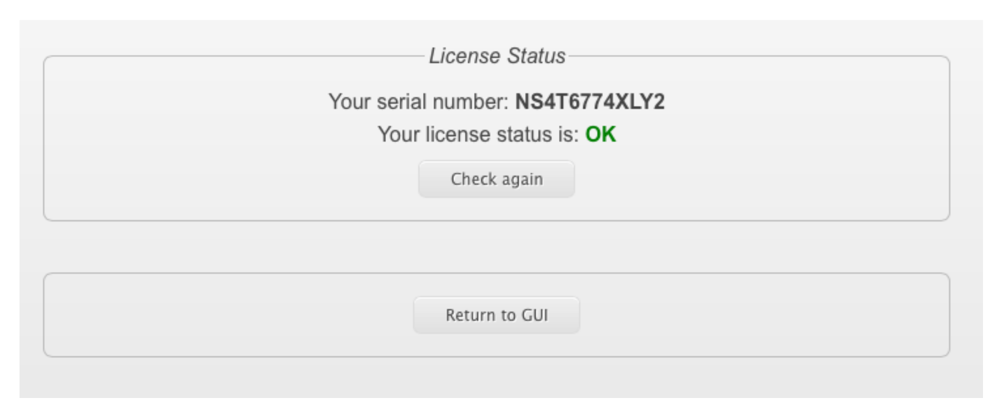
Sie öffnen den R&D Wizard durch Eingabe der IP Adresse der Box in einem Webbrowser.



Es stehen mehrere Optionen zur Verfügung die gewünschte Lizenz zu erhalten. Die Buttons „`Request Test License`" und „`Request License`" sind mit dem nomos system Shop verlinkt. Dort können Sie die gewünschte Lizenz anfordern. Folgen Sie hierzu einfach dem Dialog des Web Shop Systems.

Haben Sie eine Lizenzdatei erhalten, kopieren Sie sich die Datei zB auf den Desktop und betätigen Sie den Button „`Datei auswählen`". Verweisen Sie nun in dem Dateidialog auf das Lizenzfile.

Nach erfolgreicher Übertragung kann es es einige Sekunden dauern, bis das File validiert wurde und die Lizenz freigeschaltet ist.



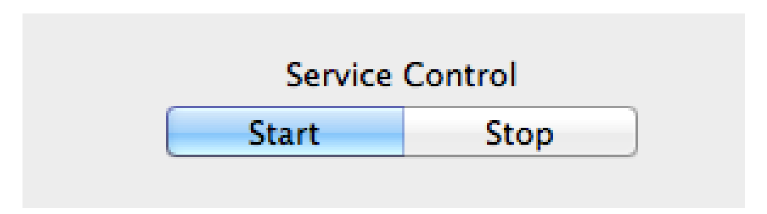
Der Status der Lizenz ist jederzeit unter der url `http://{IP-Adresse}/licensing.php` abrufbar.

2.3 Steuerung des nomos - Dienstes

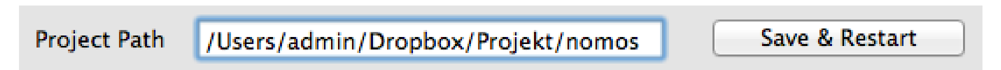
Der nomos Dienst bzw. der Daemon (nomos Box/Pi) startet automatisch, sofern eine gültige Lizenz Datei vorhanden ist. Die Mac OSX Version ist dahingegen einstellbar, ob der Dienst automatisch bei der Anmeldung aktiviert werden soll.

2.3.1 Aktivieren des Dienstes (Mac OS)

Um den Dienst zu starten, klicken Sie im Bereich „Service Control" in der nomos systemeinstellung auf „`Start`". nomos wird daraufhin unverzüglich gestartet. Wird nun im gestartetem Zustand der Button „`Save & Restart`" ausgeführt, wird das nomos system automatisch gestartet, sobald sich der Benutzer, unter dem nomos installiert wurde, am System anmeldet.



Nach der Installation ist unter „Project Path" ein Verzeichnis für die Projektdaten voreingestellt. Der dort eingegebene und individuell veränderbare Pfad auf das Projektverzeichnis muss immer vollständig angegeben werden. nomos erkennt eine Änderung des Ursprungpfades und erzeugt nach Betätigung des „Save & and Restart Buttons" eine entsprechende Verzeichnisstruktur.

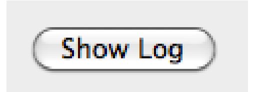


Klicken Sie im Bereich „Service Control" auf „`Stop`". Der nomos -Dienst wird unverzüglich beendet und der Systemeintrag zum automatischen Start (siehe Steuerung) entfernt.

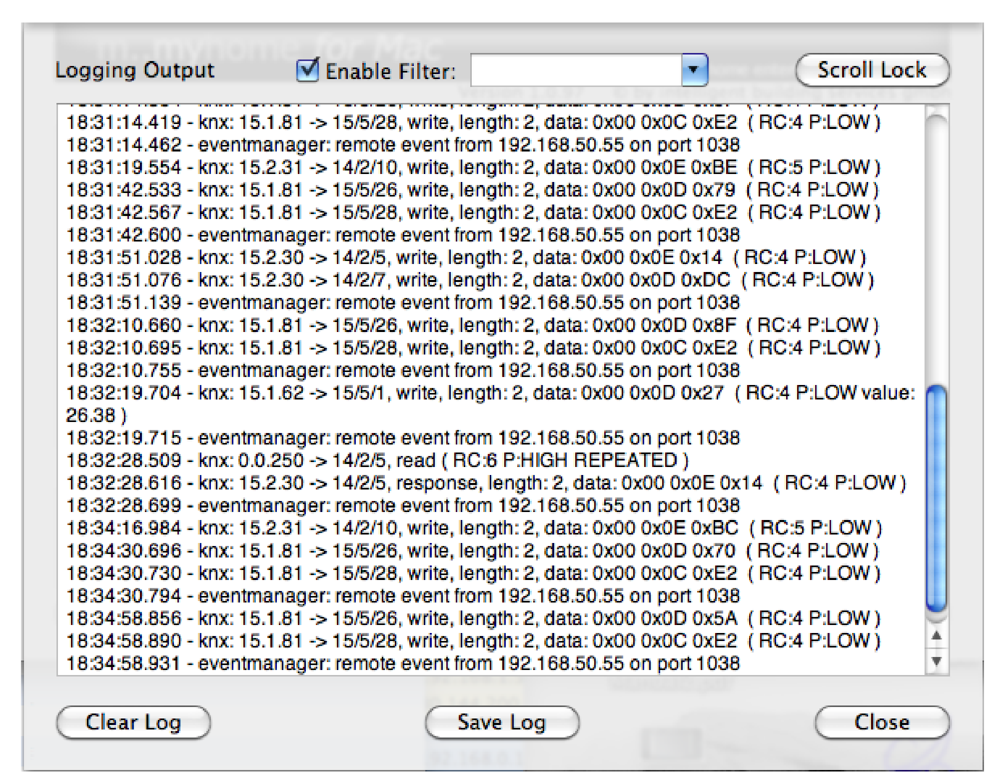
Der jeweils blau hinterlegte Teil im Bereich „Service Control" zeigt den aktuellen Status des nomos systems an.

2.3.2 Logging Monitor

nomos verfügt über einen Logging Monitor, der Auskunft über die aktuellen Zustände der jeweiligen Services gibt. Ebenso werden Fehler in den Logging Monitor geposted. Den Logging Monitor erreichen Sie durch Betätigen des Buttons „Show Log“ auf der nomos Konfigurationsseite:



Es öffnet sich der Logging Dialog. Die Ausgaben entsprechen den Ausgaben auf Port 1039.Somit sind zB fehlerhafte Einträge in den Konfigurationsdateien schnell auffindbar. Der Logging Monitor kann auch aufzeichnen und muss für diesen Zweck geöffnet bleiben. Ebenso sind Filtermechanismen eingebaut, die eine gezielte Suche erheblich vereinfachen. Über ein Pulldown Menu können vordefinierte Filter eingestellt werde. Der Eintrag kann jedoch manuell überschrieben werden. Somit sind beliebige Filteroptionen einstellbar.

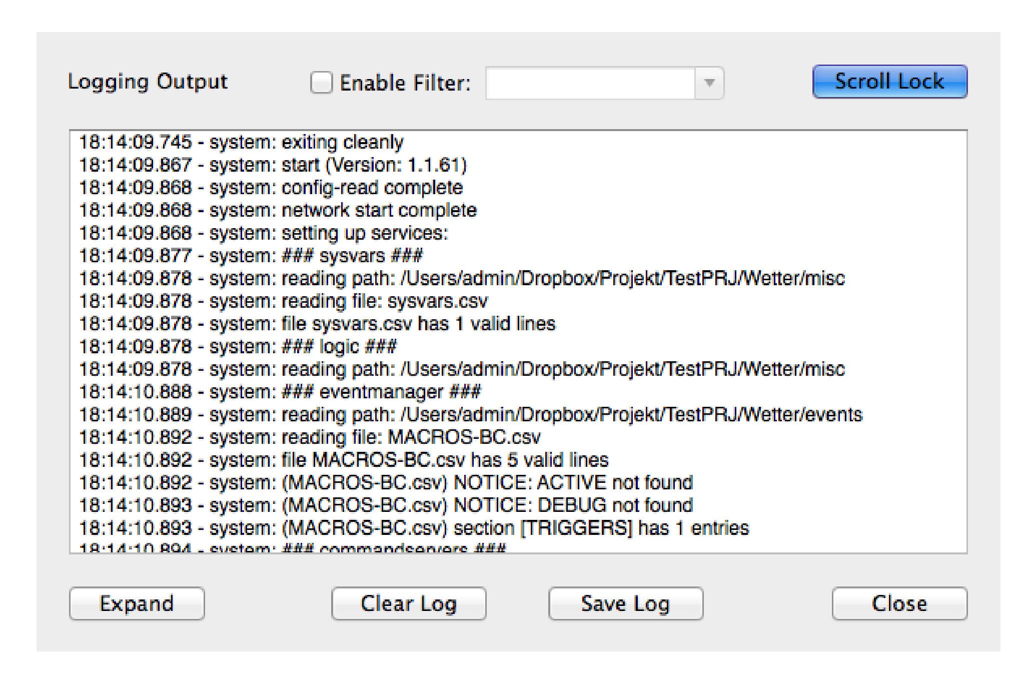


Der „`Scroll Lock`“ Button unterbindet das automatische Scrollen der Einträge. Das Log wird jedoch weitergeschrieben. Der Inhalt des Log's kann in eine Datei ausgelagert werden. Hierfür muss der Button „`Save Log`“ betätigt werden. Darauf hin öffnet sich ein File Dialog, wo der Speicherort und Name des Logfiles eingegeben werden kann. Der Button „`Clear LOG`“ löscht den aktuellen Inhalt des Fensters.

Das Logging funktioniert nur, wenn das PrefPane geöffnet ist, das Logging-Sheet selbst muß nicht geöffnet bleiben.

Das Logging erfolgt asynchron mit einem Puffer von 1000 Logzeilen

Das LOG kann grad beim Systemstart hilfreiche Informationen bieten. Das sog. Initiallog gibt Auskunft, über den Status der jeweiligen Module. Wenn innerhalb von 10 Sekunden nach dem Start des Dämons eine Verbindung zum Port 1039 hergestellt wird, wird das komplette Log ab Start ausgegeben. Es können gleichzeitig bis zu 16 Monitor Verbindungen erstellt werden



Zusätzliche Logausgaben:

Logzeilen mit ERROR oder WARNING werden jetzt rot bzw. orange dargestellt.

NOTICE bei nicht genutzten Schaltern / Sektionen in der `.csv`

WARNING bei kritischen Zuständen, die wahrscheinlich einen Fehler erzeugen werden (z.B. zu lange Zeilen)

ERROR bei notwendigen, aber nicht vorhandenen Schaltern in der `.csv`

Das Logging kann auch über NetCat aufgerufen werden. Über ein Terminalfenster kann dies local, oder aber für ein entferntes nomos system erfolgen:

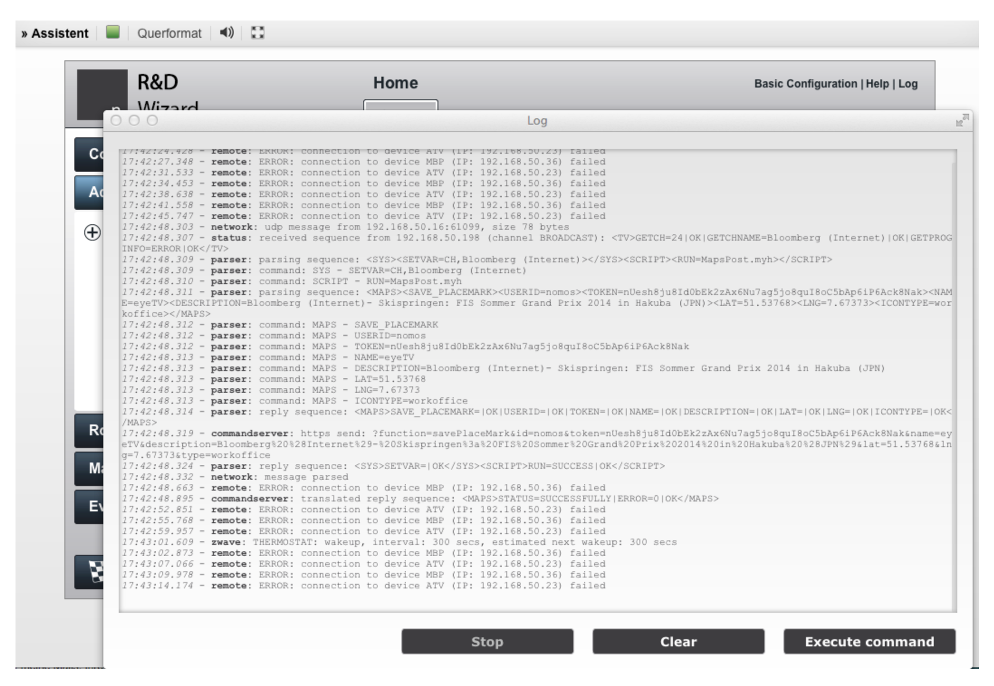
Nach der Eingabe von zB dem Befehl im Terminal Fenster:

nc 10.8.0.48 1039 Öffnet eine NetCat Verbindung auf den port 1039 der IP Adresse 10.8.0.48

```
admin — bash — 101×33
Last login: Thu Jan  2 12:37:10 on ttys000
MacBookAir:~ admin$ nc 10.8.0.48 1039
18:55:10.110 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:17.216 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:24.323 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:31.429 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:35.743 - mremote: initialising client with 2 join settings
18:55:35.743 - mremote: initialising client with 1 list settings
18:55:36.471 - mremote: initialising client with 2 join settings
18:55:36.471 - mremote: initialising client with 1 list settings
18:55:38.536 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:39.620 - mremote: action for digital join 18
18:55:39.623 - parser: parsing sequence: <KNX><SETVALUE=15/2/20,1></KNX>
18:55:39.623 - parser: command: KNX - SETVALUE=15/2/20,1
18:55:39.623 - parser: reply sequence: <KNX>SETVALUE=|OK</KNX>
18:55:39.780 - bc: <KNX><15/2/21- mike.DimBelLlp.ein/ausStatus=1></KNX>
18:55:39.907 - bc: <KNX><15/2/24- mike.DimBelLlp.WertStatus=70></KNX>
18:55:40.340 - mremote: action for digital join 14
18:55:40.343 - parser: parsing sequence: <KNX><SETVALUE=15/2/80,1></KNX>
18:55:40.343 - parser: command: KNX - SETVALUE=15/2/80,1
18:55:40.344 - parser: reply sequence: <KNX>SETVALUE=|OK</KNX>
18:55:40.529 - bc: <KNX><15/2/81- Micha.DimBelLlp.ein/ausStatus=1></KNX>
18:55:40.648 - bc: <KNX><15/2/84- Micha.DimBelLlp.WertStatus=70></KNX>
18:55:41.436 - mremote: action for digital join 17
18:55:41.439 - parser: parsing sequence: <KNX><SETVALUE=15/2/20,0></KNX>
18:55:41.439 - parser: command: KNX - SETVALUE=15/2/20,0
18:55:41.440 - parser: reply sequence: <KNX>SETVALUE=|OK</KNX>
18:55:41.725 - bc: <KNX><15/2/21- mike.DimBelLlp.ein/ausStatus=0></KNX>
18:55:41.752 - bc: <KNX><15/2/24- mike.DimBelLlp.WertStatus=0></KNX>
18:55:45.642 - remote: ERROR: connection to device MBP (IP: 192.168.50.36) failed
18:55:48.668 - bc: <KNX><14/2/8-Infozeile RTR K?che=THURSDAY,18:4514/2/8></KNX>
^C
MacBookAir:~ admin$
```

Erscheint entsprechende Ausgabe. Die Aufzeichnung kann mit Steuerung C beendet werden.

Der Wizard der nomos Box Versionen verfügt ebenfalls über ein Log Fenster. Das Logfenster des Wizards läuft in einer eigenen Instanz. Sie können also parallel zur Anzeige des Logfensters auch Änderungen und Einstellungen an den Wizard vornehmen.



Auch können Sie direkt in dem Logfenster Befehle ausführen. Betätigen Sie hierfür den Button „`Execute command`". In dem folgenden Dialogfenster geben Sie Ihre gewünschte Kommando Sequenz ein und betätigen Sie den `Send` Button.



Sie können die ausgeführte Befehlssequenz und mögliche Antwortsequenzen unmittelbar im LOG Fenster verfolgen.

2.3.3 Die nomos Verzeichnisstruktur

Die nomos Verzeichnisstruktur ist mit Ausnahme des Verzeichnisursprungs fest vorgegeben. Wie vor beschrieben, wird diese bei Installation des Services oder bei Änderung des vorgegebenen Pfades automatisch angelegt. Nachfolgend erläutern wir die einzelnen Ordner innerhalb der Projektstruktur.

Der Baum zeigt die Verzeichnisse unterhalb des Verzeichnisursprungs. In diesem Fall ist der Ursprung „nomos“. Dieser kann, wie bereits erwähnt, beliebig geändert werden.

Bei den Linux basierten Systemen ist die Verzeichnisstruktur gegenüber den Mac Systemen identisch. Einzig das Startverzeichnis `/projects` kann hier nicht verändert werden. Der Pfad `/projects/nomos` ist standardmäßig gesetzt und kann zZ nur in der Datei `/config/nomos.conf` verändert werden. Die Linux Versionen verfügen zusätzlich über eine sogenannte Config-Watch-Funktion. Der `/config` Ordner (mit `nomos.conf, nomos.key` usw.) wird permanent überwacht. Sobald sich eine Datei darin ändert, erzeugt oder gelöscht wird, startet der nomos Daemon automatisch neu.

Die jeweiligen `.csv` Dateien werden nicht automatisch generiert und sollen nur aufzeigen, wo sich die jeweiligen Dateien zu befinden haben. Zusätzlich hinzugefügte Ordner für zB Vorlagedateien sind möglich und beeinflussen in keiner Weise die Funktionalität. Bei Änderung des Projektpfades wird die Einstellung erst nach Auslösen des “Save & Restart“ Buttons aktiv.

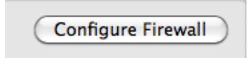
Verzeichnisstruktur

/[PROJECT]/addons	CommandServer Dateien
/[PROJECT]/events	EventServer Dateien
/[PROJECT]/gui	Button Leisten Dateien
/[PROJECT]/misc	PlugIn’s und Systemdateien
/[PROJECT]/misc/mremote/	mremote (GUI-Designer) Projektverzeichnis
/[PROJECT]/misc/web/	Bildordner für den Webserver
/[PROJECT]/misc/baos.csv	BAOS Konfiguration
/[PROJECT]/misc/hs.csv	GIRA Homeserver Konfiguration
/[PROJECT]/misc/knx.csv	KNXnet/IP Konfiguration
/[PROJECT]/misc/knx.esf	Die KNX zugehörige OPC Export-Datei. (Default Name)
/[PROJECT]/misc/logic.csv	Logik Definitionen
/[PROJECT]/misc/mremote.csv	mremote Konfiguration
/[PROJECT]/misc/remote.csv	Apple Remote Definition
/[PROJECT]/misc/sonos.csv	Sonos Player Definition
/[PROJECT]/misc/xbmx.csv	XBMC Player Definition
/[PROJECT]/misc/sysvars.csv	Systemvariabeln
/[PROJECT]/misc/timer.csv	Timer Konfiguration
/[PROJECT]/misc/webserver.csv	Webserver Konfiguration
/[PROJECT]/misc/zwave.csv	Z-Wave Kofiguration
/[PROJECT]/scripts	Verzeichnis der Scriptdateien
/[PROJECT]/XPL	Verzeichnis der xPL Konfiguration

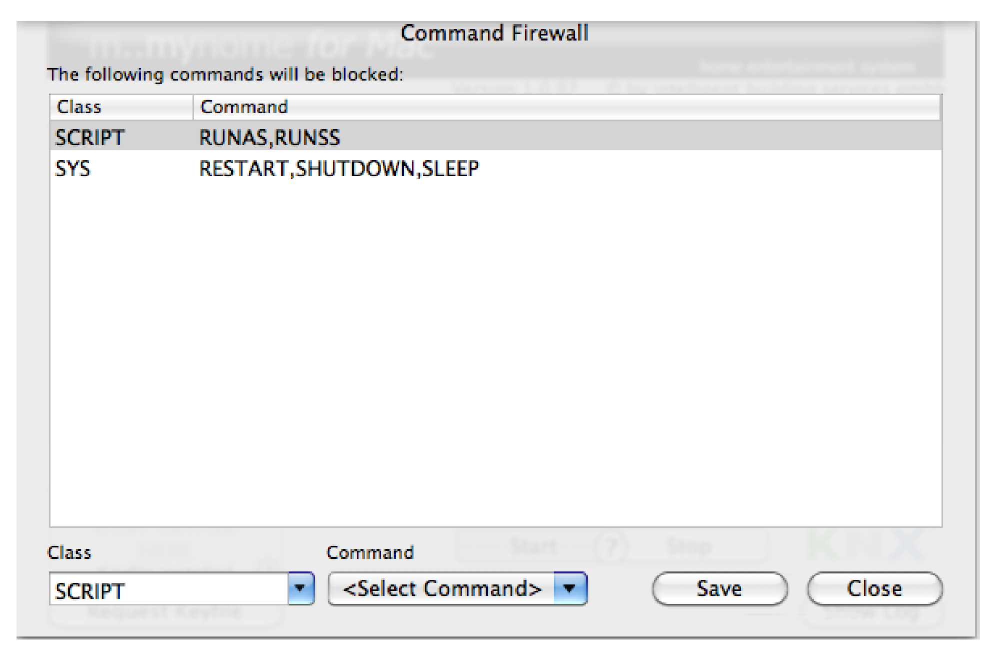
Wie die vor genannten Dateien im einzelnen aufgebaut sind, entnehmen sie den nachfolgenden Kapiteln. Das Fehlen einzelner Dateien in der Gesamtstruktur ist unproblematisch. In diesem Fall werden die zugehörigen Schnittstellen deaktiviert. Ebenso ist das Anlegen zusätzlicher Ordner für zB Vorlagedateien unproblematisch.Für komplexe Installationen wurde zusätzlich ein Verweis auf eine Projektweite `sysvar.csv` eingerichtet. Wird diese Funktion benutzt, wird die Standard `sysvars.csv` in dem `./misc` Verzeichnis ignoriert und auf die hier eingestellte Datei verwiesen. Eine genauere Erläuterung dieser Funktion finden Sie in dem Kapitel zum Umgang mit Systemvariablen.

2.3.4 Die nomos Firewall

Für zusätzliche Sicherheit und zum Schutz vor Missbrauch ist nomos mit einer eigenen Firewall ausgerüstet worden. Die Firewall ist zZ nur für die Apple OS X Versionen verfügbar.:

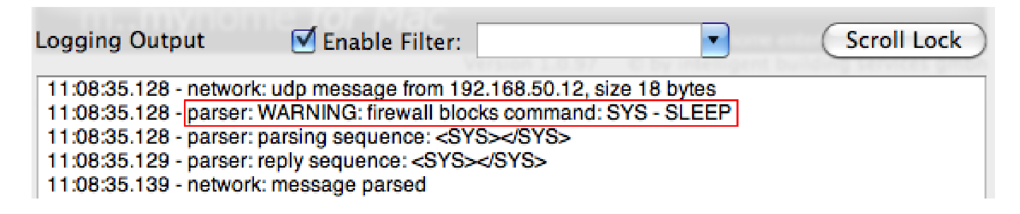


Diese Firewall schützt lediglich vor der Ausführung von Protokollanweisungen, die auf dem Zielsystem verhindert werden sollen. Man sollte sich bewusst sein, dass bei offengelegten Ports für z.B. den Zugriff über das Internet, auch Code auf das System übertragen werden und per Scriptanweisungen ausgeführt werden könnte. Ebenso könnten Befehle wie z.B. ein `SHUTDOWN` oder ein `SLEEP` für bestimmte Zielsysteme störend sein.



Für die Firewall stehen alle Befehle aller Klassen incl. der Befehle aus den eingebundenen AddOns (Command – Server) zur Verfügung. Anhängige Werte werden nicht berücksichtigt. Die Einrichtung der Firewall ist denkbar einfach. Es stehen für die Klasse und für deren Command's entsprechende Auswahlbuttons zur Verfügung. Hierüber wird der zu ignorierende Befehl einfach ausgewählt. Ausgewählte Commands sind mit einem Häckchen markiert und können entsprechend auch wieder entfernt werden. Die ausgewählten und zu ignorierenden Befehle/Commands einer Klasse werden in dem Übersichtsfenster zur Anzeige gebracht. Durch die Auswahl von `<ALL>` können alle Befehle einer Klasse markiert werden, die Auswahl `<NONE>` deaktiviert alle Befehle einer Klasse.

Im obigen Beispiel sieht man, dass die Befehle `RUNAS` und `RUNSS` aus der Klasse SCRIPT sowie die Befehle `RESTART`, `SHUTDOWN` und `SLEEP` aus der Klasse `SYS` in der Firewall aktiviert und vom nomos System ignoriert werden. Diese Einstellungen gelten jeweils nur für das Zielsystem, wo die Firewall entsprechend eingerichtet wurde. Bei mehreren nomos systemen im Netzwerk müssen die Einstellungen also je System vorgenommen werden.



Die Firewall Konfiguration wird nur beim Start von nomos gelesen. Wenn Kommandos/Klassen

blockiert werden, wird dies im Log vermerkt.

Die Änderungen werden erst gespeichert, wenn der SAVE Button betätigt wird. Nicht gespeicherte Änderungen gehen erst verloren, wenn das PrefPane geschlossen wird.

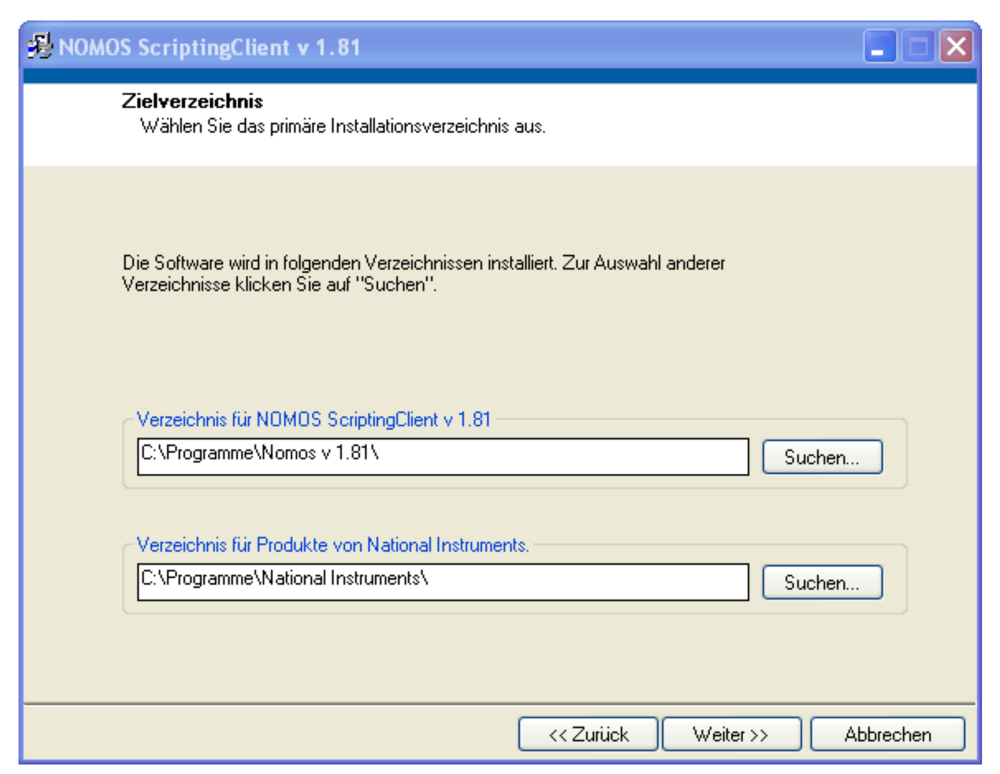
Änderungen an den Firewall Einstellungen werden erst nach einem Service Restart aktiv. CommandServer Klassen werden gelesen, wie dem System bekannt. Werden neue CommandServer Klassen eingebunden, muss auch das PrefPane geschlossen und wieder geöffnet werden, da die Klassen beim Öffnen des PrefPane einmalig gelesen werden.

2.4 Scripting Client Tool

Das Scripting Client Tool ist ein Tool, welches bei der Anwendung des nomos systems helfen soll. Mit diesem Tool können die Kommandosequenzen für die Steuerung auf einfachem Weg zusammengesetzt werden. Ebenso können die Kommandoketten als Script abgelegt werden. Diverse Logging/Monitoring Funktionen helfen bei der Suche möglicher Probleme. Das Tool wird permanent weiterentwickelt bzw. um weitere Funktionalitäten ergänzt. Diese Dokumentation wird dahingehend nicht permanent ergänzt. Zusätzliche Funktionen sollten in dem Client – Tool selbsterklärend sein. Das Tool ist für Windows und Mac OS verfügbar.

2.4.1 Installation Windows

Zur Installation der Software öffnen Sie die Datei `Setup.exe`. Es erscheint das Fenster zur Installation der Anwendung.



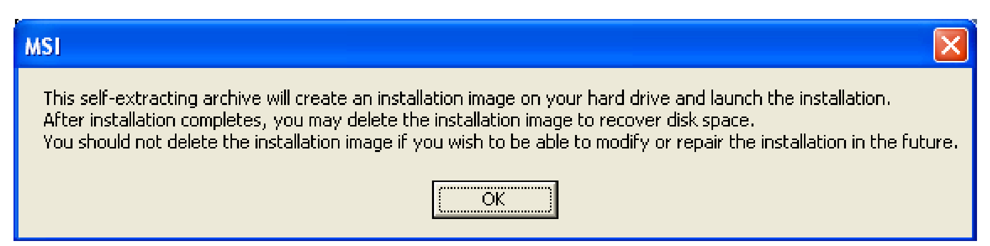
Wählen Sie nun Ihr Zielverzeichnis für die Programminstallation oder übernehmen Sie den vorgeschlagenen Pfad. Bestätigen Sie den Dialog mit „Weiter“. Im weiteren Dialog lesen Sie bitte die Lizenzbestimmungen und bestätigen Sie diese.

Nachdem Sie den Dialog mit „Weiter“ bestätigt haben, wird die Software installiert.

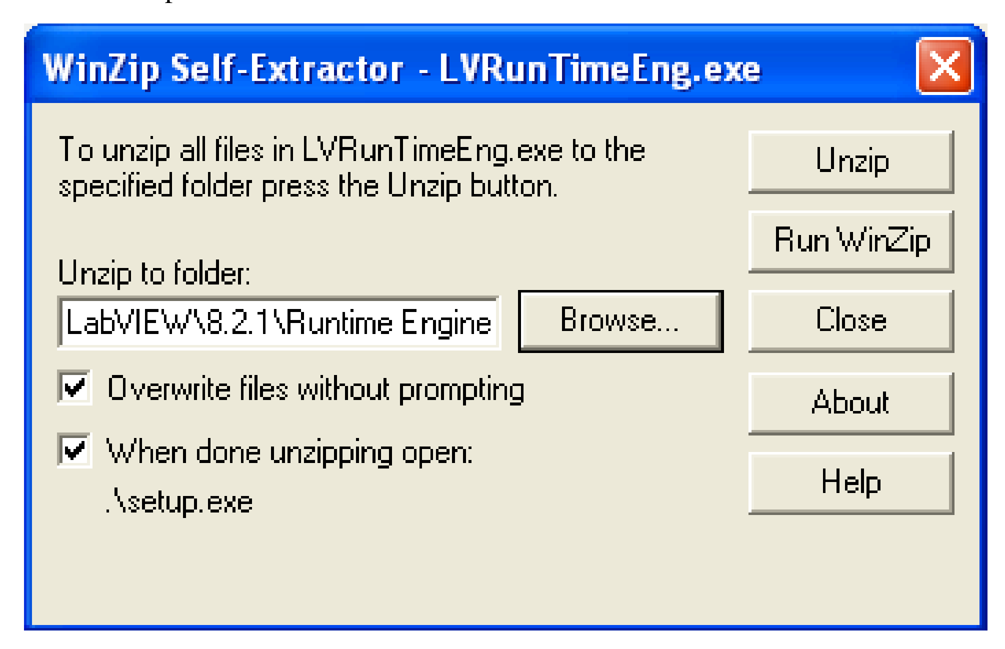
2.4.2 Installation der LabVIEW Runtime Umgebung Windows

Die Software wurde unter LabVIEW entwickelt. Dementsprechend muss die LabVIEW Runtime Umgebung installiert werden. Nutzer die bereits eine LabVIEW Runtime Umgebung auf ihr System installiert haben, können den nachfolgenden Schritt übergehen und die Installationsroutine abbrechen.

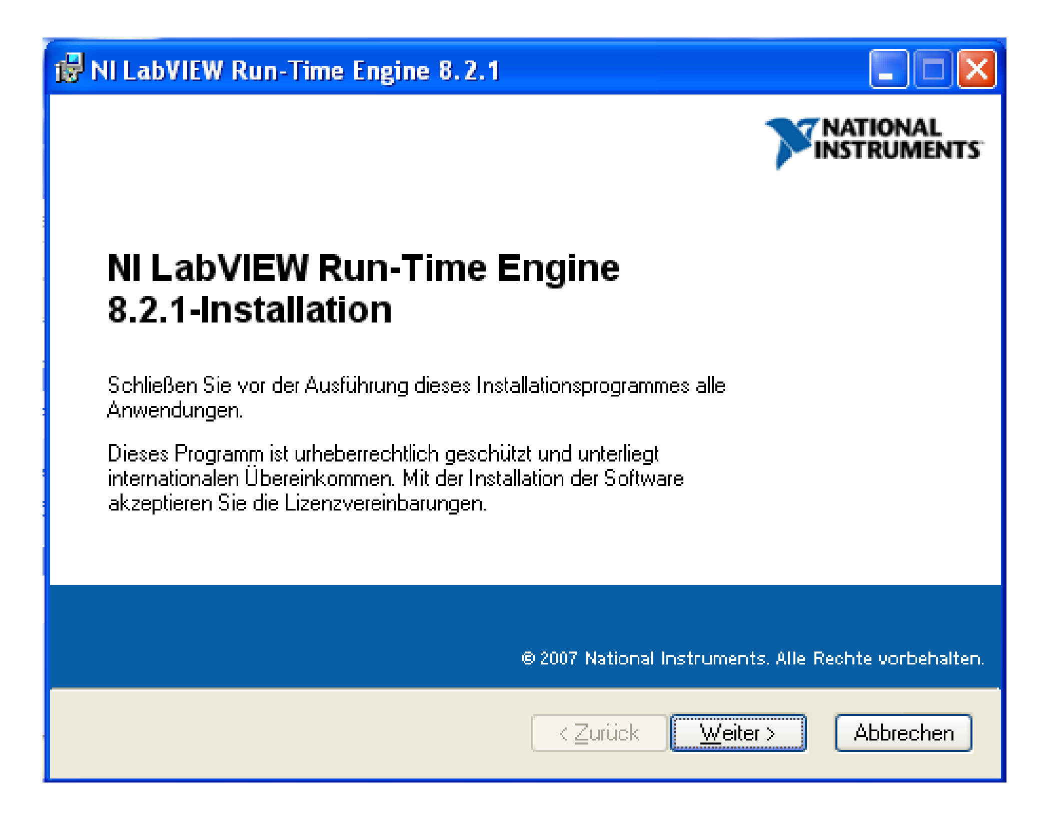
Nach erfolgreicher Installation werden Sie autom. zur Installation der LabVIEW Runtime Umgebung aufgefordert.



Bestätigen Sie das Entpacken der Archivdatei.



Bestätigen Sie den Dialog mit „Unzip“. Die Installationsdateien werden nun in ein Temporäres Verzeichnis entpackt und nach Bestätigung des nachfolgenden Dialoges wird die Installation gestartet.

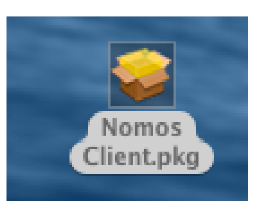


Bestätigen Sie auch diesen Dialog mit „Weiter“ und akzeptieren Sie im nachfolgenden Dialog die Lizenzvereinbarungen. Bestätigen Sie alle nachfolgenden Dialoge mit „Weiter“.

Nach erfolgreicher Installation, es erscheint ein entsprechender Hinweis, müssen Sie Ihr System neu starten. Nach erfolgtem Neustart ist der nomos Scripting Client betriebsbereit.

2.4.3 Installation Mac

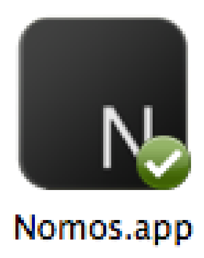
Für die Installtion auf einem Apple OSX System (ab Version 10.6) öffnen Sie bitte das Installationspaket `nomos Client.pkg`



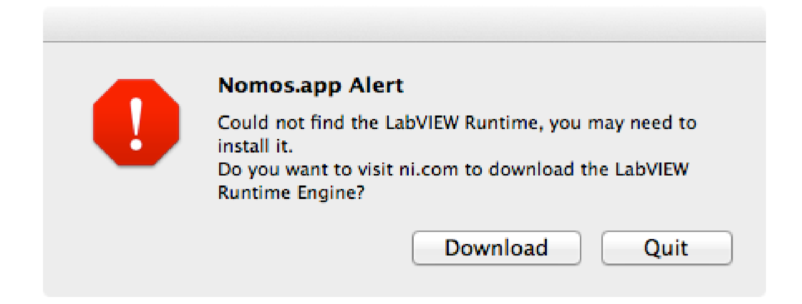
Es öffnet sich der Installationsdialog, den Sie entsprechend folgen und ausführen.



Nach Erfolgreicher Installation finden Sie die `nomos.app` Datei in Ihrem `Programme` Ordner.



Der nomos ScriptClient kann auch ohne Durchführung der Installationsroutine angewendet werden. Kopieren Sie die Datei in dem Fall manuell in den Programme Ordner und starten Sie die Applikation. Sofern die Applikation erstmalig auf einem Mac System ausgeführt wird bzw. keine LabVIEW Runtime Version vorhanden ist, erhalten Sie einen entsprechenden Hinweis.



Bestätigen Sie das Hinweisfenster durch Klicken auf den Download Button. Sie werden dann zu einer Website von National Instruments geleitet, wo Sie die Runtime Version laden können. Entsprechende Installationshinweise finden Sie dort ebenfalls.

Nach der Installation der Runtime Version können Sie die Applikation verwenden.

2.4.4 Anwendung

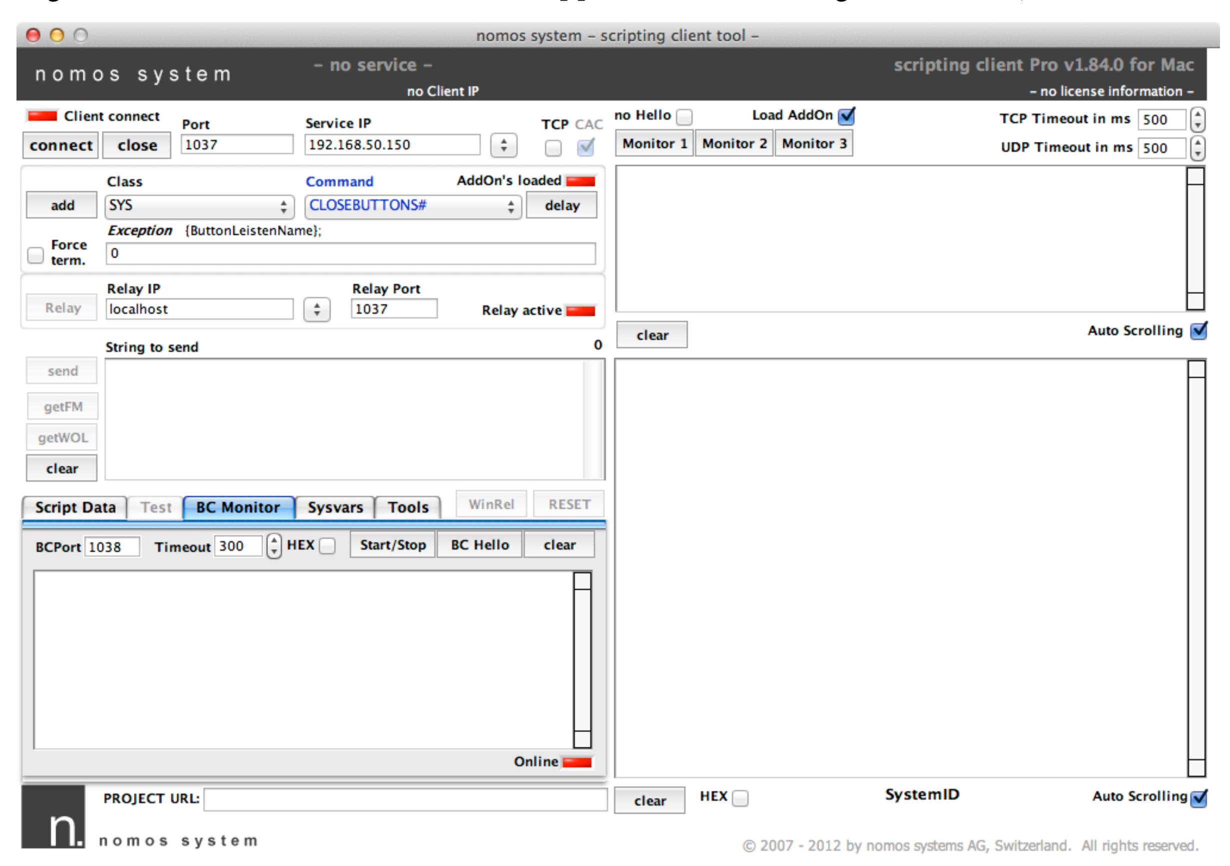
Der nomos Scripting Client dient der einfachen Erstellung von Befehlssequenzen (Script´s), die über dieses Tool generiert und getestet werden können. Ferner können die erstellten Script´s in einer internen Datenbank abgelegt werden. Das Hochladen der Scriptfiles auf Ihr nomos system kann ebenfalls über das Tool erfolgen. Ebenso ist eine einfache Schnittstelle eingebunden, die ein einfaches Implementieren neuer Befehlssätze für spätere Versionen ermöglicht.

Als Kontrollmechanismus sind Monitoring Funktionalitäten in dem nomos Scripting Client eingebunden worden. Dies ermöglicht eine Überwachung der gesamten Kommunikation. Detaillierte Informationen entnehmen Sie dem Kapitel Monitoring.

Den nomos Scripting Client starten Sie durch einen Doppelklick auf das Programm Icon:



Es erscheint nachfolgendes Programmfenster. (Windows: Es befindet sich auch ein entsprechender Eintrag im Startmenü - Mac: Die Datei `nomos.app` befindet sich im Programme Ordner)



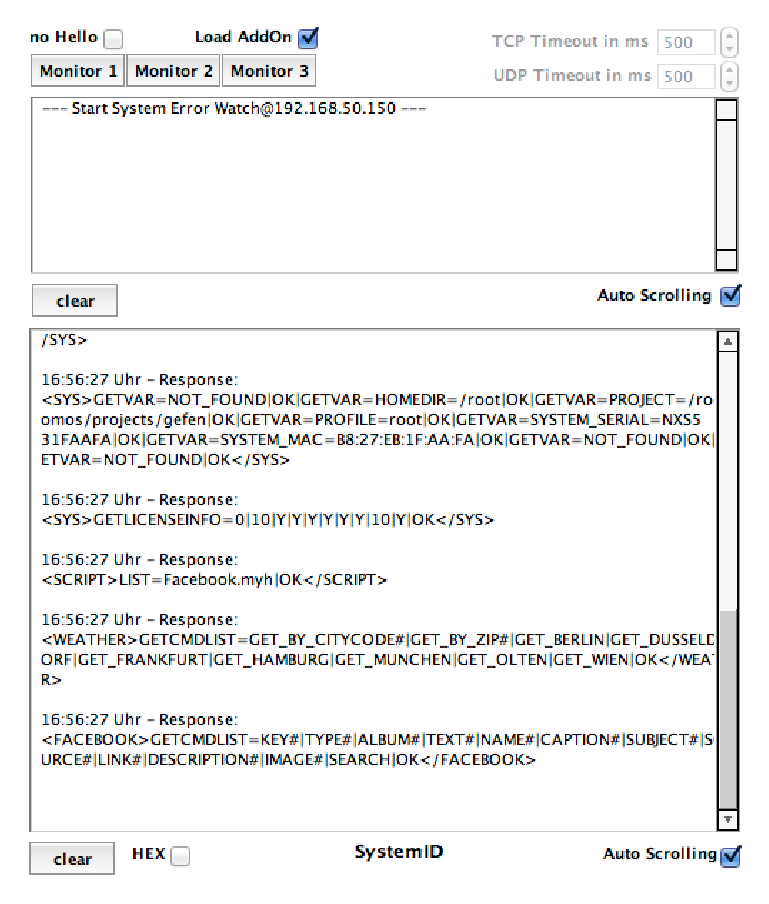
Sofern Sie alle vorherigen Schritte erfolgreich ausgeführt haben, können Sie nun eine Verbindung mit Ihrem nomosSystem aufbauen. Geben Sie hierfür die IP Adresse und den entsprechenden Port (Standardeinstellung ist `1037`) ein. Sie können zwischen einer TCP oder UDP (Standard) Verbindung wählen. Beachten Sie hierbei, dass Sie die Einstellungen konform mit den Einstellungen auf Ihrem nomos System vornehmen.

Wenn Sie alle Eingaben vorgenommen haben, betätigen Sie den Button `Connect`. Die Software merkt sich alle IP Adressen, mit der eine erfolgreiche Verbindung aufgebaut werden konnte. Wollen Sie einen Eintrag aus der Liste entfernen, so wählen Sie den gewünschten Eintrag aus der Liste, positionieren Sie den Mauszeiger über den IP-Adresseintrag und drücken Sie die rechte Maustaste. Es erscheint ein Kontextmenü mit dem Inhalt „`Adresse entfernen`". Markieren Sie den Eintrag im Kontextmenü mit dem Mauszeiger und betätigen Sie zur Bestätigung die linke Maustaste.



Ist die Verbindung aufgebaut wird dies durch eine Statusanzeige bestätigt. Ebenso wird die aktuell installierte nomos Version des verbundenen Systems, sowie deren MAC-Adresse angezeigt.

Im Monitoring Fenster können Sie den Datenaustausch beobachten.



Nach dem Kommunikationsaufbau fordert der nomos ScriptingClient durch die Befehlsketten `<SYS><HELLO></SYS>` die aktuelle nomos Programmversion des verbundenen nomos systems an und durch `<SCRIPT><LIST></SCRIPT>` die Liste der auf dem verbundenen System gespeicherten Scriptfiles. Näheres hierzu wird im nachfolgenden Verlauf des Dokumentes erklärt.

Ebenso werden Informationen über die installierte Lizenz sowie die MAC Adresse des verbundenen Apple Systems angezeigt.

Hinweis:

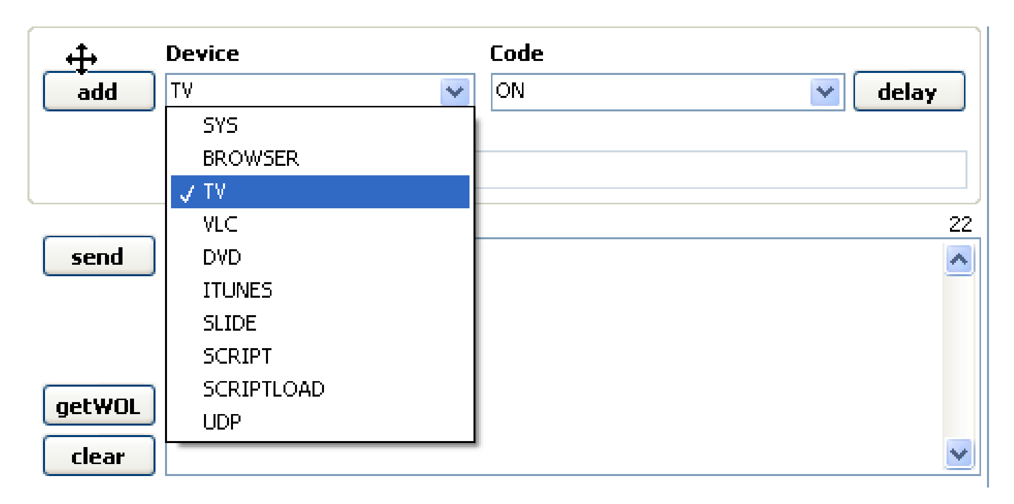
Der Scripting Client aktiviert sich durch eine bestätigte Verbindung mit dem nomos Service. Ohne bestätigte Verbindung kann der Scripting Client keine TCP/UDP Pakete versenden.

2.4.5 Umgang mit dem ScriptingClient

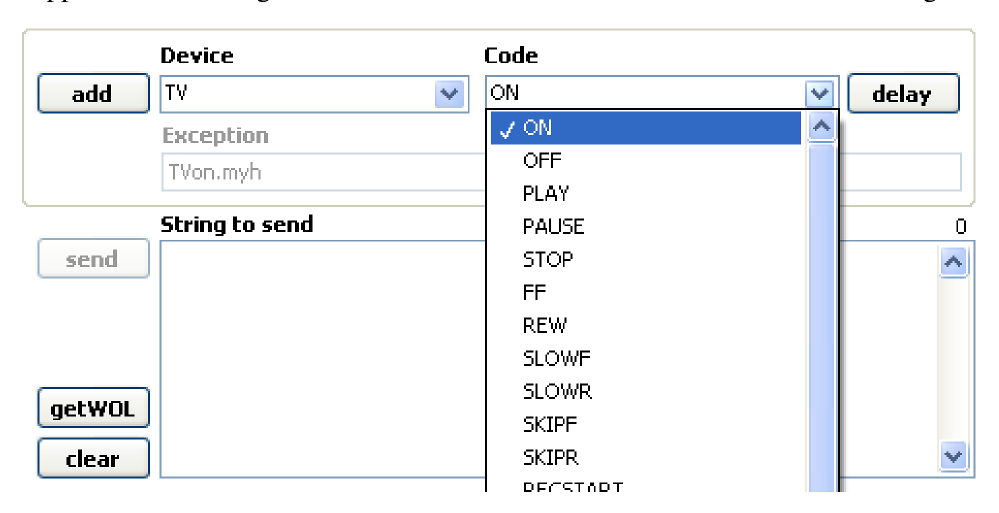
Nachdem Sie nun eine erfolgreiche Verbindung zu Ihrem nomos system aufbauen konnten, können Sie beginnen Befehle an das nomos system zu senden. Verwenden Sie hierbei immer eine plausible Kette von Befehlen und beachten Sie, dass die Befehle sequentiell abgearbeitet werden. Dies bedeutet, dass Sie möglicherweise erst ein Gerät einschalten müssen, bevor es weitere Befehle entgegen nehmen kann. Gehen Sie hierbei also genau so vor, wie Sie zB Ihre Anwendungen lokal an dem Apple Computer bedienen würden oder aber ein externes Gerät, zb ein TV Gerät, mittels Fernbedienung steuern. Im nachfolgenden Beispiel wird das Steuern der eyeTV-Anwendung auf einem Apple Computer erklärt. Nähere Beschreibung zu den möglichen Befehlen sowie deren Funktionen entnehmen Sie dem Kapitel Befehlssatz.

Der nomos Befehlssatz gliedert sich in Klassen (Device), dem Klassenbefehl (Code) und einen evtl. zu übergebenden Wert (Exception). Die codespezifischen Eigenschaften bzw. der notwendige Aufbau der zu sendenden Sequenz wird Ihnen durch den Scripting Client komplett abgenommen. Sie wählen lediglich die Befehle, die Sie senden möchten, über die Auswahlfelder aus und fügen Sie der Sequenz hinzu.

Um nun die Anwendung eyeTV zu öffnen, wählen Sie wie nachfolgend abgebildet die Klasse (Device) `<TV>`.

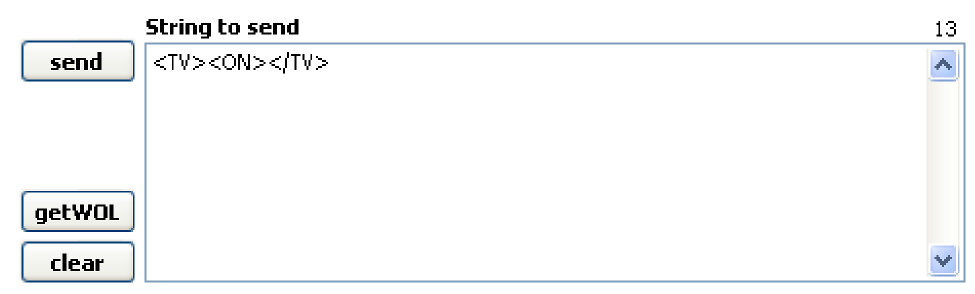


Nach Auswahl der Klasse können Sie sich die verfügbaren Klassenbefehle anzeigen lassen und den gewünschten Befehl auswählen. Das Öffnen einer Applikation erfolgt grundsätzlich mit dem `ON` Befehl. Sofern die Applikation bereits geöffnet war, hat der `ON` Befehl keine weitere Auswirkung.



Nachdem Sie nun die Klasse (Device) `<TV>` sowie den Befehl (Code) ausgewählt haben, betätigen Sie

den Button „add“. In der „String to send“ Anzeige erscheint nun der vollständige Befehl, der an den nomos Service gesendet wird.



Betätigen Sie nun den „send“ Button um den String an das nomos system zu senden. Auf dem Apple Computer startet nun die eyeTV-Applikation, sofern diese installiert wurde. Ist Ihr System ohne TV-Empfang eingerichtet verwenden Sie eine andere Klasse, z.B. `<ITUNES>` oder `<DVD>`.

Nach dem Versand der Befehlskette wird im Monitoring Fenster die erfolgreiche Kommunikation angezeigt. Der nomos bestätigt (Response) jeden Befehl mit einen „`|OK`“. Kann ein Befehl nicht interpretiert werden, erscheint ein „`|NOK`“.

11:37:36 - Send:
<TV><ON></TV>

11:37:36 - Response:
<TV>ON=|OK</TV>

Eine erweiterte Logging-Funktion ist ebenfalls gegeben und wird im nachfolgendem Verlauf des Dokumentes ausführlicher beschrieben.

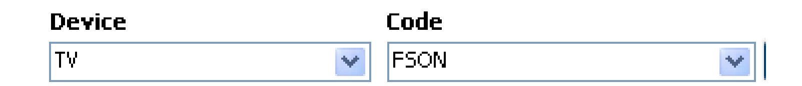


Wenn Sie den Mouse Zeiger über den aktuellen Befehl bewegen, erscheint ein Tooltip mit Angabe, welche Funktion der Befehl beinhaltet.

Sie werden feststellen, dass weitere Einstellungen notwendig sind, um die Anwendung in den gewünschten Zustand zu bringen. In der Regel öffnen sich die Anwendungen auf dem Apple Computer nicht immer im Vollbildmodus, wie es für z.B. bei der TV oder DVD Wiedergabe gewünscht ist. Ferner werden weitere Einstellungen benötigt, wie evtl. die Vorgabe der Lautstärke, die Vorgabe der Programmwahl, das Schließen einer vorherig gestarteten Anwendung.

Im nachfolgenden Beispiel werden wir den vorherig erklärten Befehl entsprechend komplettieren, so dass die TV-Applikation entsprechend im Vollbildmodus mit definierter Lautstärke und definierter Kanalwahl startet. Bei der Reihenfolge der Befehle gehen Sie bitte ebenso vor, wie Sie es bei der lokalen Bedienung an dem Apple Computer einstellen würden.

Ausgangsposition ist unser String: `<TV><ON></TV>`
Diesen String erweitern wir nun um den Befehl `FSON` (Vollbildmodus).



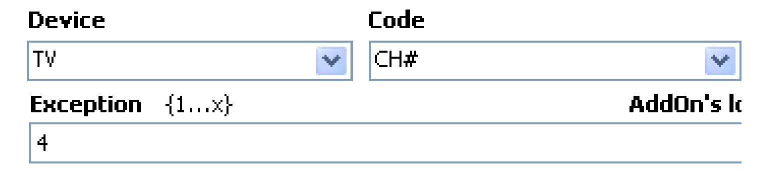
Wählen Sie entsprechend Device und Code aus und betätigen Sie den add- Button. Beachten Sie bitte, dass die ausgewählten Befehle immer am Ende der Befehlskette angefügt werden. Soll der String an anderer Position eingefügt werden markieren Sie bitte vor dem Betätigen des add-Buttons die gewünschte Position im „String to send“ mit dem Mauszeiger und bestätigen Sie die Position durch einen Klick auf die linke Mousetaste. Danach betätigen Sie die add-Button und der Befehl fügt sich an ausgewählter Position im „String to send“ ein. Bitte beachten Sie, dass nun intern die einfüge Position „gemerkt“ wird. D.h. alle weiteren Befehle werden immer hinter dem zuletzt eingefügten Befehl angereiht; bis Sie die Einfügeposition, wie vor beschrieben, erneut ändern. Sie können ebenfalls händisch diesen String verändern, Einträge oder Wertinhalte löschen bzw. erweitern. Nachdem Sie nun den „add“ Button betätigt haben, sollte der String wie folgt aussehen:

<TV><ON><FSON></TV>

Oder bei Verwendung von iTunes versuchen Sie:

<ITUNES><ON><WINSIZE=FULL></ITUNES>

Nun erweitern wir den String um die Kanalvorwahl. Ohne Kanalvorwahl startet eyeTV grundsätzlich mit dem zuletzt eingestellten Kanal. Suchen Sie nun in den Befehlen (Code) der Klasse (Device) <TV> den Befehl CH#. Wenn ein Befehl mit dem Sonderzeichen „#“ endet bedeutet dies, dass zusätzlich ein Wert erwartet wird, der unter „Exception“ eingetragen werden muss. Dementsprechend wird das Eingabefeld aktiviert. Der Wert, der erwartet wird, wird hinter „Exception“ angezeigt. Für den Befehl CH ist dies ein Wert zwischen 1 und der maximalen Anzahl der vorhandenen TV-Programme, die eyeTV zur Verfügung stellt.



Tragen Sie nun den Wert Ihrer Wahl in das Feld „Exception“ ein und betätigen Sie den add-Button. Danach sollte der String wie folgt aussehen:

<TV><ON><FSON><CH=4></TV>

Für iTunes verwenden Sie den PLAYPLAYLIST Befehl um die Auswahl einer Playliste zu starten:

<ITUNES><ON><WINSIZE=FULL><PLAYPLAYLIST=Musik></ITUNES>

Sie können immer wieder mal den String senden um die Auswirkungen auf dem Apple Computer zu verfolgen. Sie werden so auch feststellen, dass „doppelt“ gesendete Befehlsketten keine störenden Auswirkungen auf das laufende System haben.

Sollte bei Ihnen die Applikation „eyeTV“ oder „iTUNES“ vor dem versenden des Strings geschlossen gewesen sein, wird das erstmalige Senden dieser Befehlskette lediglich die Applikation öffnen. Dies liegt daran, dass die Applikation erst empfangsbereit ist, wenn sie vollständig gestartet und entsprechend bereit ist. Dies dauert i.d.R. wenige Sekunden. Jedoch hat der nomos Service die anhängigen Befehle längst an die Applikation gesendet. Da die Applikation aber noch nicht „bereit“ war, liefen die Befehle ins Leere. Sie haben hier die Möglichkeit, die Ausführung der Befehle gezielt zu verzögern. Dies wird nachfolgend noch genauer erklärt.

Im nächsten Schritt erweitern Sie nun den Befehlssatz um die Systemlautstärke, mit der die Anwendung starten soll. Wählen Sie hierfür nun in der Klasse SYS den Befehl VOLSET aus. Dieser Befehl erwartet wieder einen zusätzlichen Wert (Exception). Tragen Sie in das Feld einen beliebigen Wert zwischen 0 und 100 ein.

Nach dem Betätigen der „add“ Taste sollte der String wie folgt aussehen:

<TV><ON><FSON><CH=3></TV><SYS><VOLSET=30></SYS>

Für iTunes entsprechend:

<ITUNES><ON><WINSIZE=FULL><PLAYPLAYLIST=Musik></ITUNES><SYS><VOLSET=30>
</SYS>

Es ist völlig egal, an welcher Steller die Zeichenkette `<SYS><VOLSET=30></SYS>` nun erscheint, da es keine Auswirkung auf das Ergebnis hat, ob Sie vor dem Öffnen der Applikation oder danach eine Änderung der Lautstärke vornehmen.

Man kann nun noch den String vervollständigen in dem man z.B. Eventualitäten wie einen aktivierten Bildschirmschoner, oder ein vorheriges „Muten“ der eyeTV-Applikation berücksichtigt und durch entsprechend aufhebende Befehle ergänzt.

Wählen sie hierfür in der System Klasse `SYS` den Befehl `DISABLESS` und betätigen Sie den add-Button und wählen Sie im Anschluss aus der Klasse `TV` den Befehl `MUTEOFF` und betätigen Sie erneut den add-Button

Der String sollte nun wie folgt aussehen:

<TV><ON><FSON><CH=3></TV><SYS><VOLSET=30><DISABLESS></SYS><TV><MUTEOFF>
</TV>

Für iTunes entsprechend:

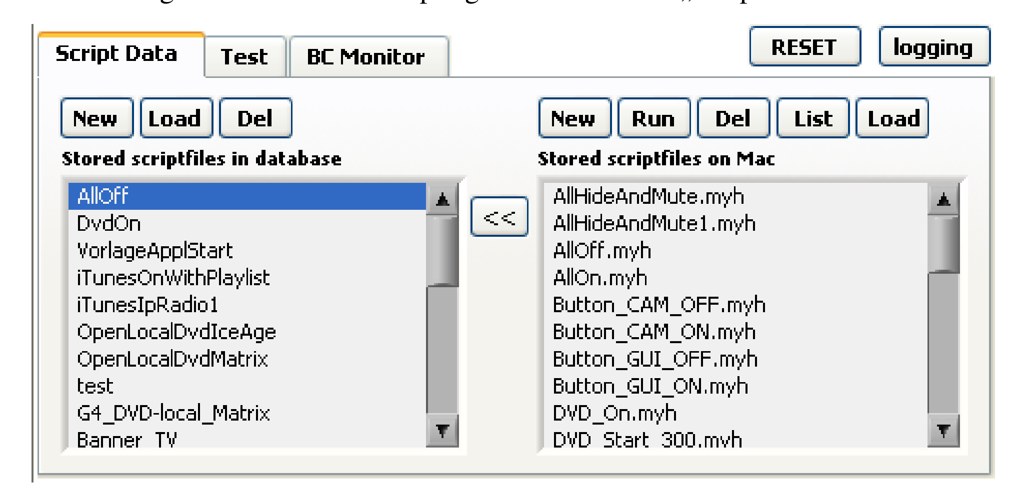
<ITUNES><ON><WINSIZE=FULL><PLAYPLAYLIST=Musik></ITUNES><SYS><VOLSET=30>
<DISABLESS></SYS><ITUNES><MUTEOFF></ITUNES>

Wenn Sie diesen Befehl nun an den Apple Computer senden, wird die Applikation entsprechend im Vollbildmodus mit der Kanalvorwahl `3`, einer Systemlautstärke von `30%` starten. Ein evtl. aktivierter Bildschirmschoner wird deaktiviert, ebenso wie ein evtl. vorheriges Applikation Mute.

Nun können Sie die gesamte Befehlskette als Script auf dem Apple Computer hinterlegen bzw. in die interne Script Datenbank ablegen.

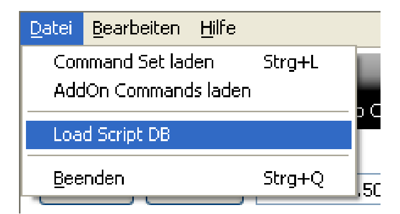
Gehen Sie hierbei wie folgt vor:

Wählen Sie in der Registerauswahl des Scripting Client die Karte „Script Data“



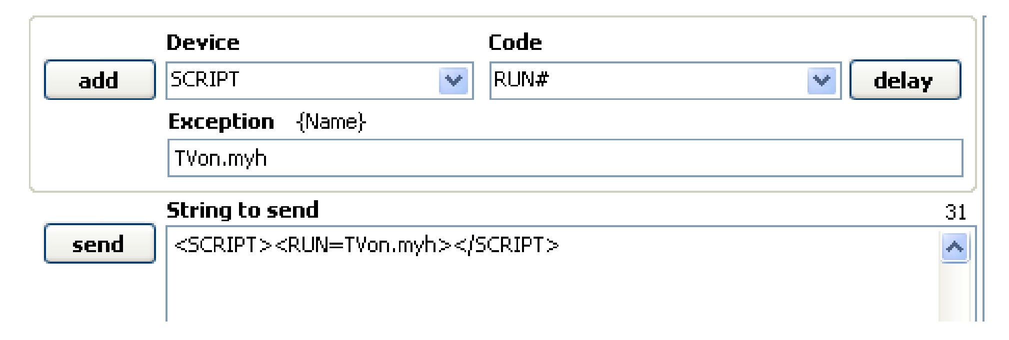
Im linken Teil der Karte befinden sich die Script – Files der internen Script-Datenbank und im rechten Teil der Karte befinden sich die ausführbaren Scripts, wie sie auf dem System abgelegt wurden. Wenn Sie nun den String als Script speichern möchten, betätigen Sie den Button „New“ und geben Sie einen Namen für das Scriptfile ein (Bspw. `TVon` oder `iTunesON`). Die Dateiendung (Suffix) wird automatisch behandelt und ist mit `.myh` fest vorgegeben.

Ist das Scriptfile auf dem System gespeichert, wird es in der rechten Auswahltabelle entsprechend angezeigt. Scripts können jederzeit zur Bearbeitung in das „String to send“ Fenster geladen werden. Markieren sie hierzu das entsprechende Scriptfile durch einen Mausklick und betätigen Sie den „Load“ Button. Möchten Sie das ausgewählte Script löschen, betätigen Sie den „Del“ Button und möchten Sie es ausführen (hierbei muss das Script auf dem System vorhanden sein), betätigen Sie den „Run“ Button.



Grundsätzlich sollten die Scripts auch immer in der internen Script-DB gespeichert werden. Sie können mehrere interne Datenbanken anlegen bzw. auswählen. Die Datenbank ScriptDB.sdb ist die default Datenbank, die beim Programmstart standardmäßig geöffnet wird. Es empfiehlt sich z.B. projektbezogene Datenbanken anzulegen. Spätere Versionen werden ein direktes kopieren des Datenbankinhaltes auf dem System unterstützen.

Scripte können jedoch auch über die Klasse `<SCRIPT>` gestartet werden. Wählen Sie entsprechend die Klasse wie nachfolgend dargestellt aus:

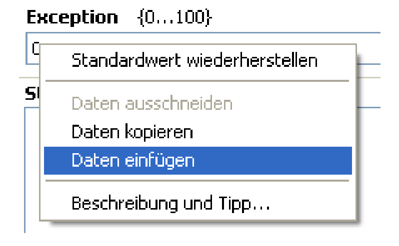


Bitte Achten Sie stets darauf, dass bei dem Dateinamen Gross- und Kleinschreibung sowie das Suffix berücksichtigt werden muss.

Sie können den Namen des markierten Scriptfiles über die Zwischenablage übernehmen indem Sie durch klicken mit der rechten Maustaste das Kontextmenu öffnen und „copy to clipboard“ bestätigen.



Unter „Exception“ öffnen Sie dann ebenfalls durch einen Rechtsklick das Kontextmenu und bestätigen „Daten einfügen“.



Die Daten der Zwischenablage können ebenso in anderen Dialogen verwendet werden. Ebenso ist dieses Verfahren fast überall anwendbar.

Mit Ihrem ersten erstellten Script haben Sie nun eine Funktion, die wie vorher beschrieben, Ihre TV-Anwendung in einem definierten Zustand bringt. Nun möchten Sie aber nicht für jeden einzelnen Pro-

grammplatz ein Scriptfile erstellen. Nachfolgendes Beispiel zeigt Ihnen, wie Sie dynamische Argumente (z.B. den Programmplatz/Kanal) dem Scriptfile beim Aufruf übergeben können.

Laden Sie hierzu das Scriptfile wieder in das „Script to send“ Fenster (`TVon.myh` oder `iTunesON.myh` markieren und Load- Button betätigen). Ändern Sie den String wie folgt ab:

> <TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=30></SYS>

Für iTunes entsprechend:

> <ITUNES><ON><WINSIZE=FULL><PLAYPLAYLIST=[ARG]></ITUNES>
> <SYS><VOLSET=30><DISABLESS></SYS><ITUNES><MUTEOFF></ITUNES>

Speichern Sie das Script erneut ab. Verwenden Sie hierbei einen schlüssigen Namen, wie z.B. `TVon(CH).myh` oder `iTunesON(PL).myh`. Wir verwenden diese Art der Beschreibung um zu erkennen, dass das Scriptfile ein Argument, nämlich `CH={Kanalnummer}` oder `PL={Playliste}`, benötigt.

Generieren Sie nun den Aufruf Code wie folgt:

> <SCRIPT><RUN=TVon(CH).myh><ARG=5></SCRIPT>

Für iTunes entsprechend:

> <SCRIPT><RUN=iTunesON(PL).myh><ARG=Musik></SCRIPT>

Senden Sie den String nun zu dem Apple Computer und Sie sehen, dass die TV-Anwendung mit dem gewünschten bzw. vorgegebenen Programm bzw. die iTunes Anwendung mit der gewünschten Playliste öffnet. Sie können beliebig viele Argumente übergeben. Möchten Sie nun auch die Lautstärke vorgeben ist folgende Änderung notwendig.

Ändern Sie erneut Ihr Scriptfile wie folgt ab:

> <TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=[ARG]></SYS>

Für iTunes entsprechend:

> <ITUNES><ON><WINSIZE=FULL><PLAYPLAYLIST=[ARG]></ITUNES><SYS><VOLSET=[ARG]>
> <DISABLESS></SYS><ITUNES><MUTEOFF></ITUNES>

Speichern Sie das Scriptfile z.B. unter den Namen `TVon(CH,VOL).myh` bzw unter `iTunesON(CH,VOL).myh` und generieren den Aufruf Code:

> <SCRIPT><RUN=TVon(CH,VOL).myh><ARG=3><ARG=50></SCRIPT>

oder entsprechend:

> <SCRIPT><RUN=iTunesON(CH,VOL).myh><ARG=Musik><ARG=50></SCRIPT>

Ändern Sie die Argumente nach belieben und senden Sie den String erneut. Sie werden sehen, dass neben dem Programmplatz sich nun auch die Lautstärke einstellen lässt.

Sie können Argumente beliebig verwenden. Argumente können auch anstelle von URL’s, Bildpfade, Musiktitel, Fenstergrößen oder auch Code oder Klassen verwendet werden. Das Argument ersetzt oder ergänzt eine Kommandozeile um den jeweiligen Wertinhalt/Text bevor der Aufruf abgearbeitet wird.

Argumente können auch global gefüllt werden.

Ein Script mit dem Inhalt <[ARG]><ON></[ARG]> kann durch Verwendung des Übergabebefehls `GARG` global gefüllt werden. Folgender Aufruf würde alle Platzhalter mit dem übergebenen Wert füllen:

> <SCRIPT><RUN=Name(Class).myh><GARG=ITUNES></SCRIPT> -> <ITUNES><ON></ITUNES>

ARG= und GARG= können auch kombiniert werden. Hierbei ist zu beachten, dass erst ARG= zu verwenden ist um gezielt Platzhalter zu füllen. Bei Verwendung eines anschließenden GARG= werden alle restlichen Platzhalter gefüllt.

<TV><ON><FSON><CH=[ARG]></TV><SYS><VOLSET=[ARG]></SYS><[ARG]><ON></[ARG]>

Das Script kann mit folgendem Aufruf entsprechend abgehandelt werden:

<SCRIPT><RUN=Name(CH,VOL,ClassOFF).myh><ARG=3><ARG=50><GARG=ITUNES
></SCRIPT>

> Entsprechend würde das 1. `ARG` mit „3“, das 2. `ARG` mit „50“ und das 3. und 4. `ARG` mit „ITUNES“ gefüllt werden, also wie folgt dargestellt. (Die Abarbeitung der Argumente erfolgt stets sequentiell.):

<TV><ON><FSON><CH=3></TV><SYS><VOLSET=50></SYS><ITUNES><ON></ITUNES>

Im Wesentlichen haben Sie das Prinzip des ScriptingClient nun kennengelernt. Es empfiehlt sich die jeweiligen Befehle einmal nach und nach zu testen um die Auswirkungen auf dem zu steuernden nomos System kennenzulernen.

Natürlich können die Scriptdateien auch manuell erstellt und in das Verzeichnis `.\scripts` abgelegt werden. Die `.myh` Dateien sind einfache ASCII Dateien, die mit jedem Texteditor erstellt und bearbeitet werden können.

2.4.6 Scripte und deren Anwendung

Wie vor beschrieben definieren Scripte einen kompletten Funktionsablauf durch die Aneinanderreihung einzelner Befehle. Die Verwendung von Scripte vereinfacht die Konfiguration eines nomos systems erheblich. Ferner muss bei der Änderung einer Script-Definition während der Laufzeit kein Reset durchgeführt werden, da ein auszuführendes Scriptfile immer erneut eingelesen wird. Ebenso können dynamische Inhalte bei einem Scriptaufruf (Call) übergeben werden. Wir bevorzugen daher grundsätzlich in den nachfolgend beschriebenen Konfigurationsdateien die Verwendung von Scriptaufrufen anstelle der Verwendung von reinen Command-Ketten.

Natürlich können auch „Script in Script" Aufrufe ausgeführt werden. Hierbei ist zu beachten, dass durch die sequentielle Abarbeitung Verzögerungen eintreten können.

Beispiel:

`Script0.myh` mit Inhalt:
`<SCRIPT><RUN=Script1.myh><RUN=Script2.myh><RUN=Script3.myh></SCRIPT>`

Die Ausführung würde mit folgendem Befehl eigeleitet:
`<SCRIPT><RUN=Script0.myh></SCRIPT>`

Entsprechend würden nacheinander `Script1.myh` bis `Script3.myh` ausgeführt werden. Je nach Inhalt der jeweiligen Scripte wird die Ausführung erst fortgesetzt, sofern das laufende Script abgearbeitet wurde.

Eine Parallele Abarbeitung kann mittels dem Befehl FORCERUN erzwungen werden. Hierbei wird das Script eingelesen und in einem eigenen Thread abgearbeitet:

> <SCRIPT><FORCERUN=Script1.myh><FORCERUN=Script2.myh><FORCERUN=Script3.myh>
> </SCRIPT>

Der Aufruf mittels Befehl <SCRIPT><RUN=Script0.myh></SCRIPT> würde nun alle drei Scriptfiles einlesen und in einem eigenen Thread zur parallelen Ausführung ablegen.

Reservierte Script-Namen:

Ein zusätzliches Feature sind die sog. reservierten Scripts. Diese werden automatisch ausgeführt, sofern vorhanden. Mit diesen Scripts können Sie u.a. das Startverhalten von nomos beeinflussen.

init.myh Der Inhalt des Scripts wird nach einem Servicestart ausgeführt.

exit.myh Der Inhalt des Scripts wird vor Beenden des Services ausgeführt

sleep.myh Der Inhalt des Scripts wird vor der Standby-Funktion ausgelöst

wakeup.myh Der Inhalt des Scripts wird nach dem „Aufwecken" ausgelöst

Nachfolgend einige Beispiele für die Verwendung dieser reservierten Scripte:

Das `init.myh` Script:

Sofern vorhanden und der nomos Service auf autostart konfiguriert wurde, wird dieses Script nach dem Systemstart automatisch ausgeführt. Hier können z.B.

durch entsprechenden Aufruf innerhalb des Scriptes: Applikationen automatisch gestartet werden; „Welcome Messages“ zur Anzeige gebracht werden ; Parameter wie System- und/oder Applikationslautstärken vorbelegt werden. Ebenso könnte z.B. auch bei Systemstart das angeschlossene TV oder Audio System automatisch eingeschaltet werden.

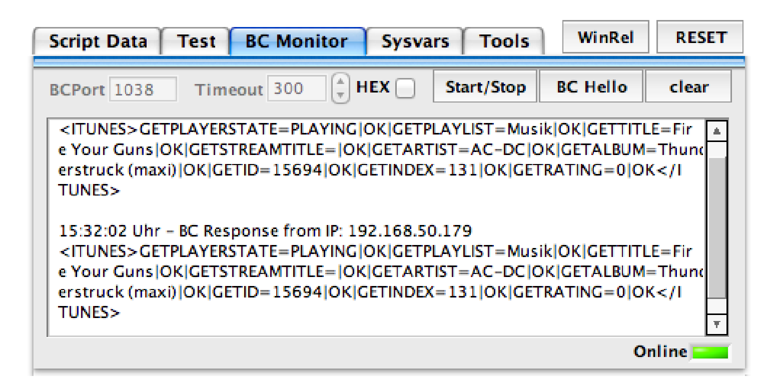
Das `exit.myh` Script:

Wird ausgeführt, wenn der laufende nomos Service beendet wird. Dies gilt auch, wenn das nomos system über die Systemsteuerung heruntergefahren wird. Hier können z.B. Ausschalt- Befehle für die weitere Peripherie wie eben das TV- oder Audiosystem abgelegt werden.

2.4.7 Broadcast Monitoring

Der nomos Scripting Client verfügt neben dem bereits erwähnten Monitoring-Fenster auch über einen Broadcast-Monitor und einem System-Log-Monitor. Der Broadcast-Monitor dient zur Ansicht der Meldungen, die nomos über den Broadcast Status-Port in das Netzwerk überträgt.

Um den Broadcast-Monitor zu starten, wechseln Sie auf die Karteikarte „BC Monitor“.



Betätigen Sie den Button „Start“ und ändern Sie über Ihre Tastatur z.B. die Lautstärke. Sie werden sofort eine Feedback-Message mit entsprechend den aktuellen Werten erhalten. Ebenso sendet nomos Infos zum aktuell laufenden Musiktitel oder dem aktuell laufenden TV-Programm. Auch der PLAYER STATUS der jeweiligen Anwendungen wird übertragen; für iTunes bspw. ein `PLAYED` oder `PAUSED`.

Broadcast-Statusmeldungen können auch über den Eventserver empfangen und ausgewertet werden um wiederum entsprechende Aktionen auszulösen. Der Eventserver wird im weiteren Verlauf des Dokumentation noch genauer beschrieben

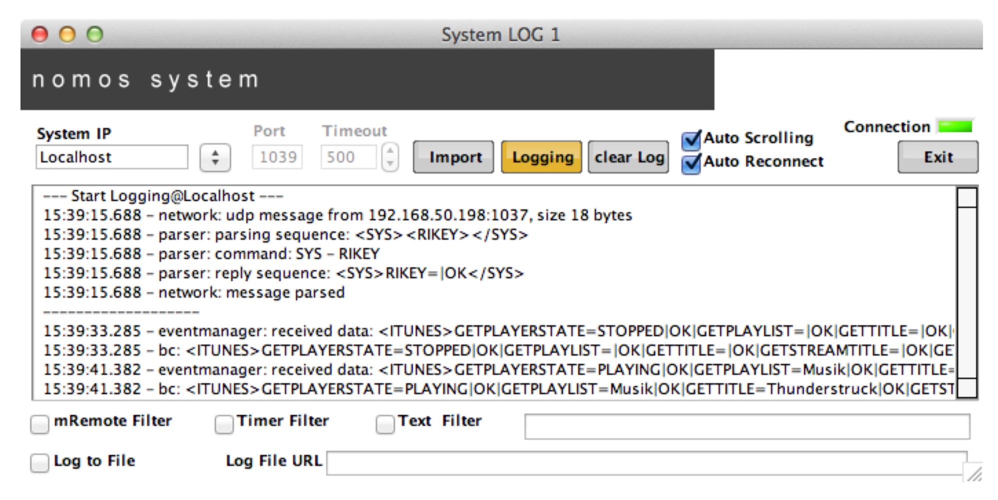
System Monitor

Weitere Monitoring-Fenster öffnen sich durch Betätigen der „`Monitor n`“ Tasten.

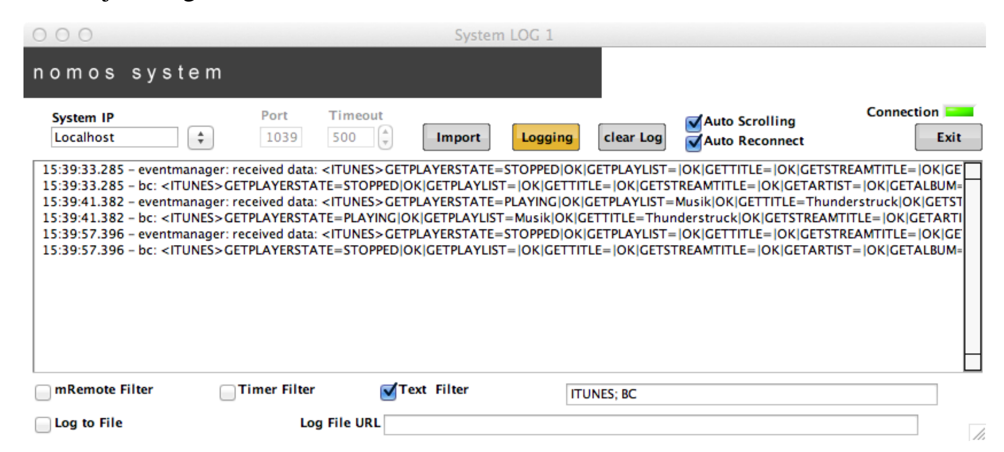


Dies ist der Systemmonitor, der Auskunft darüber gibt, was der nomos Service abarbeitet. Dieser Monitor dient z.B. der Analyse sofern Probleme eintreten bzw. kann unplausibles Verhalten aufzeigen. Ebenso können die Sendesequenzen bei Verwendung der `CommandServer-` Funktion beobachtet werden.

Der Monitor muss durch Betätigung des Logging-Buttons gestartet werden. Der Port `1039` ist fest eingestellt und kann nicht verändert werden.



Der Monitor verfügt über einige Filterfunktionen, die den Umgang etwas vereinfachen. Mehrere Filterkriterien können ebenfalls angegeben werden. Hierbei ist ein Semikolon als Trennzeichen für die jeweiligen Filter anzuwenden.



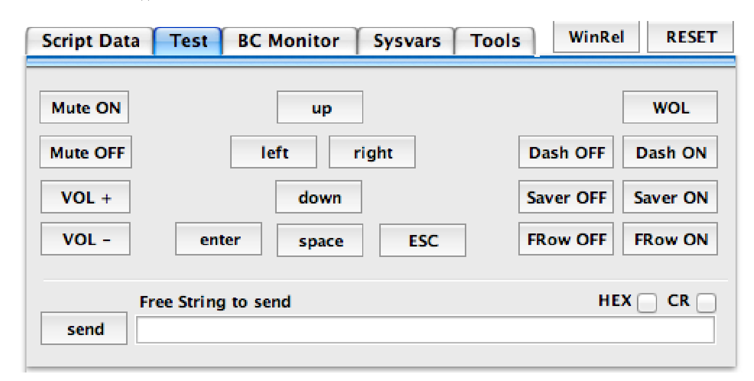
Ebenso kann das LOG in eine Datei Exportiert und entsprechende LOG Files auch wieder importiert werden.

Benutzen Sie diese logging Funktion vor allem um das Startlog zu beobachten. Die Startsequenz wird ca. 30s nach Start des Services gepuffert und beim Öffnen des Port 1039 komplett ausgegeben. Im Startlog ist erkennbar, ob alle Module ordnungsgemäss gestartet oder Fehler aufgetreten sind

2.4.8 Testfeld

Um grundsätzliche Steuerfunktionen schnell zu testen, wurde ein kleines Testfeld in den ScriptingClient eingebunden.

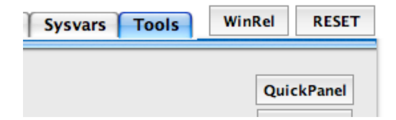
Das Testfeld finden Sie unter der Karteikarte „Test“



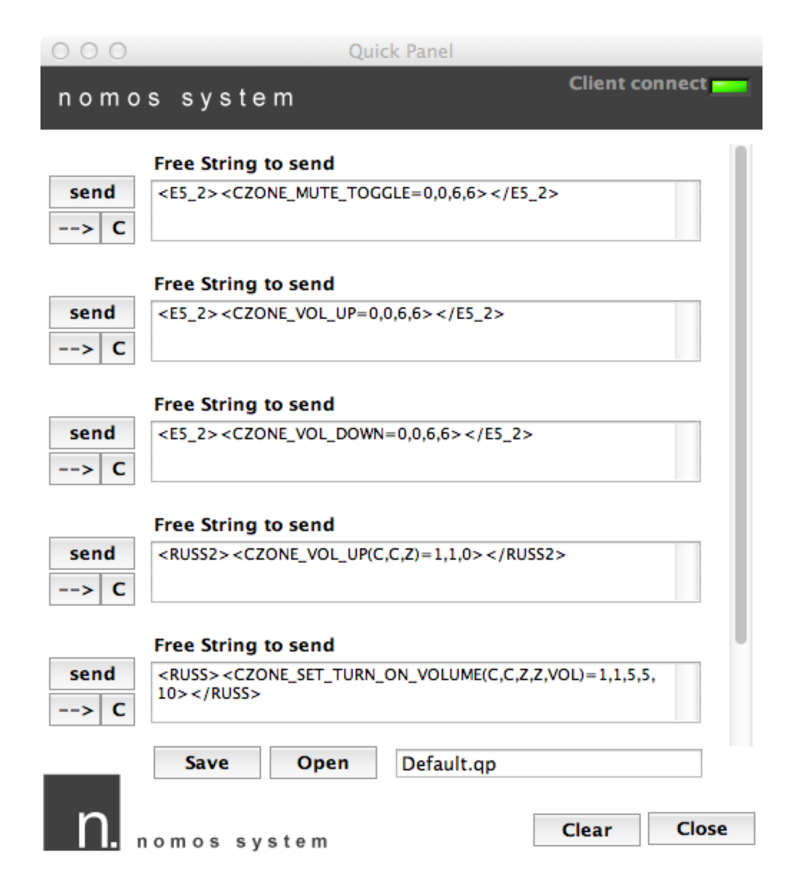
Die Buttons sind mit spezifischen Systemfunktionen vorbelegt. Sobald der Client mit Ihrem nomos system verbunden ist, lassen sich die Funktionen ausführen. In dem Feld „Free String to send“ können freie Befehlsketten an das System gesendet werden. Die Funktionen der jeweiligen Buttons entnehmen Sie den Hinweisstreifen (Tipstrip/ Mouseover).

Benutzen Sie den „Free String to send“ um z.B. auch häufig verwendete Befehlsketten oder Script Aufrufe zu „parken“.

Eine weitere Hilfe finden Sie unter der Karteikarte „Tools“, das QuickPanel.



Es öffnet sich ein weiteres Dialogfenster. Hier können Sie beliebig viele Kommando- Sequenzen ablegen und wie eine Arbeitsvorlage abspeichern und auch wieder einlesen. Zum Einlesen der .qp Datei reicht es, die Datei über das nomos Logo zu ziehen (Drag&Drop)



Die Kommando Sequenzen können via Drag&Drop Funktion aus dem Hauptdialog Fenster direkt in das Quick Panel gezogen werden. Ebenso werden die Daten aus der Hauptanwendung bei Betätigung des „–>” Buttons übernommen.

2.4.9 Weitere Funktionen

Im Dateidialog des nomos ScriptingClient finden Sie einige weitere Funktionen. Diese beinhalten neben dem bereits beschriebenen Laden der Script-DB auch eine Funktion zum aktualisieren des Befehlssatzes (Command Set) sowie den Abruf evtl. vorhandener Command-Server Definitionen (AddOn).



Sobald Ihr System über entsprechende Definitionstabellen verfügt, wird der Befehlssatz bei Abruf der Daten erweitert. Hinweise zu dem `CommandServer-` Dateien finden Sie im weiteren Verlauf der Dokumentation.

Mit der Funktion „Command Set laden" können Sie den aktuellen Befehlssatz aus der Definitionsdatei `CommandList.csv` einlesen. Die aktuellste Definitionsdatei finden Sie auf unserer Webseite www.nomos-system.com

KAPITEL 3

Der nomos Befehlssatz

Nach dem Start des nomos systems bindet nomos auf allen im System verfügbaren Netzwerk-Schnittstellen, die in der Konfiguration definierten Ports an sich und wartet auf eingehende Verbindungen. Erhält nomos ein Kommando, das dem definierten Kommandosatz entspricht und eine gültige Syntax aufweist, führt er dieses Kommando aus und meldet die erfolgreiche Ausführung sowie ggfs. auftretende Rückgabewerte zurück an den aufrufenden Client. Die Kommunikation aller angebundenen Systeme erfolgt hierbei über eine einheitliche und durchgängige Kommando Struktur, dem nomos system Protokoll.

Je nach Art des Kommandos werden hierbei unterschiedlichste Aktionen im System ausgelöst. Der Umfang reicht von einfachen, die Benutzeroberfläche des Systems betreffenden Aktionen bis hin zum kompletten Abschalten bzw. Neustarten des Systems.

Das nomos system bietet ebenfalls die Möglichkeit, verschiedene Scripte und Befehlssequenzen lokal abzuspeichern und bei Bedarf vollautomatisch auszuführen.

3.1 Begrifflichkeiten / Typen

Der Befehlssatz der nomos - Software untergliedert sich in zwei Befehlstypen

3.1.1 Klassen (CLASS)

Klassen `<{CLASS}>` dienen dazu, nomos mitzuteilen, welcher Befehlstyp ausgeführt werden soll. Außerdem wird so der unterstütze Befehlssatz in einem überschaubaren, strukturierten Rahmen gehalten. Weiterhin müssen Klassenaufrufe immer durch `</{CLASS}>` terminiert werden (siehe Aufbau des Protokolls), sodass mit diesem Verfahren gleichzeitig eine Sicherung gegen unbeabsichtigt unterbrochene Verbindungen und unvollständige Sequenzen gewährleistet ist. Ein Klassenaufruf führt noch keine Befehle im nomos system aus.

3.1.2 Kommandos (CMD)

Kommandos `<{CMD}>` stellen den eigentlichen Befehlssatz des nomos systems zur Verfügung. Sie können eigenständig oder mit Argumenten versehen sein.

3.1.3 Scripte

Ein Script ist eine Kette von Kommandos, dass als aufrufbare Funktion zur Verfügung steht (Makro). Es können beliebig viele Scripte erzeugt und abgerufen werden. Ferner können dem Script auch bis zu 10 unterschiedliche Argumente beim Aufruf übergeben werden. Ein Script kann während der Laufzeit programmatisch erstellt , verändert oder auch gelöscht werden.

3.1.4 Sonderkommandos

Sonderkommandos zb `<DELAY=n>` sind klassenunabhängig und können an beliebigen Stellen in der Sequenz eingefügt werden. Es gelten die syntaktischen Regeln der Standardkommandos. Es wird jedoch empfohlen, Sonderkommandos außerhalb einer Klasse zu verwenden bzw. unmittelbar nach einer Terminierung (`</{CLASS}>`).

3.1.5 Formatierungsoptionen (optional)

Formatierungsoptionen können optional den jeweiligen Kommandos angefügt werden. Über entsprechende Formatstrings können Ausgaben beliebig beeinflusst werden. Ebenfalls können hierüber Such- und Sortierfunktionen eingeleitet werden.

3.1.6 Konverter bzw. Konvertierungsoptionen (optional)

Konvertierungsoptionen können optional verwendet werden um Argumente zu beeinflussen, umzurechnen oder Umzuwandeln.

3.2 Syntax (Aufbau des Protokolls)

Array Abarbeitung

<CLASS><CMD></CLASS>
Die Syntax für einen gültigen Befehl **ohne Argument**
Beispiel: `<SYS><RESTARTSERVICE></SYS>`

<CLASS><CMD=ARG></CLASS>
Die Befehlssequenz für einen Befehl **mit Argument,**
wobei `ARG` für das einzusetzende, kommandoabhängige Argument steht.
Beispiel: `<SYS><SETVOL=24></SYS>`

<CLASS><CMD{FORMATSTRING}></CLASS>
Die Befehlssequenz für einen Befehl **Formatierungsoptionen:**
wobei der optionale {FORMATSTRING} für den entsprechend einzusetzenden Ausdruck steht.
Beispiel: `<ITUNES><GETALLPLAYLISTS{%l3,5}></ITUNES>`

<CLASS><CMD=ARG{KONVERTERSTRING}></CLASS>
Die Befehlssequenz für einen Befehl **mit Argument und Konvertierungsoptionen:**
wobei der optionale {KONVERTERSTRING} für den entsprechend einzusetzenden Ausdruck steht. Formatierungs- und Konvertierungsoptionen können auch gleichzeitig innerhalb eines Kommandos Anwendung finden.
Beispiel: `<SYS><SETVOL=0A{%dec,%+30,%/2}></SYS>`

<CLASS><CMD1><CMD2><CMD3=ARG1><CMD4>......<CMDn></CLASS>
Kommandos lassen sich aneinanderreihen, so dass eine zusammenhängende Befehlssequenz entsteht.
Beispiel: `<ITUNES><SHOW><FRONT><PLAYPLAYLIST=Musik></ITUNES>`

<CLASS1><CMD1></CLASS1><CLASS2><CMD2></CLASS2><.....></CLASSn>
Auf dieselbe Weise lassen sich verschiedene Klassen in einer Sequenz kombinieren.
Beispiel: `<TV><ON></TV><SYS><SETVOL=44></SYS>`

<SYS><SETVAR=TEST,<SYS><SAY=HELLO></SYS>></SYS>
Ebenfalls ist das Senden verschachtelter Sequenzen möglich.

Auch lassen sich `1Byte` Hexwerte in die Kommandosequenz einbinden. Hierüber können z.B. auch nicht sichtbare Zeichen übergeben werden. Die Übergabewerte werden in die entsprechende dezimale Zahl gewandelt. Hexadezimale Werte werden durch ein vorangestelltes „`\0x`" kenntlich gemacht.

Beispiel:

```
<SYS><SETBUTTONLABEL=main,2,Zeile 1\0x0d Zeile 2></SYS>
```

Schreibt einen zweizeiligen Text in die Hauptbuttonleiste.

Die maximale Länge einer `Befehlssequenz` beträgt `8192 Byte` (Zeichen). Wird eine Klasse nicht terminiert, so werden alle in diesem Klassenaufruf enthaltenen Befehle verworfen.

3.2.1 Array Abarbeitung

Das nomos system Protokoll unterstützt ebenfalls Array's. Hiermit können z.B. Listen verwendet werden. Ein Array definiert sich durch das Trennzeichen „`|`". Hierüber ist es z.B. auch möglich, Mehrfachabfragen zu generieren. Wird z.B. einem Kommando als Argument eine solche Liste übergeben, wird intern diese Liste nacheinander abgearbeitet und eine entsprechende Antwortsequenz erzeugt. Dieses Verhalten ist gleichzusetzen mit einer „FOR-Schleife“, wie man sie aus der Informatik kennt.

Beispiel:

```
<ITUNES><GETIDTITLE=1080></ITUNES>
```

Würde als Antwort-String beispielsweise folgende Ausgabe generieren:

```
<ITUNES>GETIDTITLE=One Of The Brightest Stars|OK</ITUNES>
```

Wird nun aber eine Liste mit mehreren Titeln übergeben wie z.B.:

```
<ITUNES><GETIDALBUM=1040|1046|1052|1058|1062|1064|1066></ITUNES>
```

Erfolgt die Ausgabe entsprechend:

```
<ITUNES>GETIDTITLE=Carry You Home|Give Me Some Love|I'll Take
Everything|One Of The Brightest Stars|Same Mistake|OK</ITUNES>
```

Intern zerlegt das nomos system die übergebene Liste in deren einzelne Elemente und führt entsprechend ein `GETIDALBUM` mit dem jeweiligen Element aus. Das Ergebnis wird wiederum als Liste ausgegeben.

3.3 Antwortsequenz

Die Antwortsequenz des nomos - System besteht aus einer Wiederholung der Kommandosequenz. Argumente werden nicht wiederholt. Zwecks XML-Kompatibilität wird bei den Kommandos auf die umfassenden „<" und „>" verzichtet. Mehrere Kommandos derselben Klasse werden durch „|" getrennt. Syntaktisch gültige Kommandos werden in der Antwortsequenz immer mit dem Argument „OK" versehen, ungültige Kommandos mit „NOK":

<CLASS1>CMD1=OK|CMD2=NOK</CLASS1><CLASS2>

Im obigen Beispiel war CMD1 ein gültiges Kommando; CMD2 war ungültig.

Enthält die Antwort auf ein Kommando ein Argument (ARG), wird das Argument zuerst hinter dem Befehl eingefügt: <CLASS>CMD=ARG|OK</CLASS>

Ebenso verhält es sich bei der Verwendung von Formatoptionen. Die Parameter werden ebenfalls der Antwortsequenz angefügt: <CLASS>CMD{FORMATSTRING}=ARG|OK</CLASS>

Die maximale Länge einer Antwortsequenz kann 8129 Byte (Zeichen) betragen.

3.4 Formatieroptionen, Parser Optionen, Status Port

Nachfolgend beschreiben wir verwendete Platzhalter für diverse Funktionalitäten, wie dem Einleiten von Abfragen mit formatierter Ausgabe, dem Auswerten bzw. Parsen von Textinformationen sowie der Informationsübergabe. Formatoptionen/Konvertieroptionen beziehen sich jetzt immer auf das voranstehende Teilargument.

3.4.1 Formatieroptionen

Verwendung von Formatoptionen, die formatierte Rückgabewerte erzwingen können. Dies ist zB notwendig um Timeticker, die in Sekunden ausgegeben werden, automatisch in Minuten und Stunden zu formatieren. Ebenso können hierüber gezielte Listenausgaben generiert werden, wie sie z.B. bei einer iTunes-Playlisten Abfrage notwendig sind.

Die Option wird an den Befehl, deren Rückgabewert ausgewertet werden soll, in „`{}`" eingefasst angehängt.

Folgende Formatierungsoptionen sind z.Z. verfügbar:

%h	rechnet einen Wert in volle Stunden um
%m	rechnet den Wert in volle Minuten um
%s	stellt den Wert 1:1 dar
%l	(Liste) initiert eine nach „Index“, „Länge“ indizierte Ausgabe
%lc	(count) Zählt die Elemente einer Liste (sollte am Ende in der Optionskette stehen)
%ls	(sort) Sotiert eine Liste alphabetisch
%lf	(filter) Filtert die Ausgabe der Liste
%lu	(unique) Filtert alle mehrfach vorhandenen Elemente aus
%c	(Teilstring) gibt einen Teilstring aus
%r	(Ersetzen) ersetzt eine Zeichenkette durch eine Neue
%t	Dummy, um z.b Listenausgaben zu bennen (Tagging)
%fxml	Formatierbefehl für xml-Content (s. unten)

Formatoptionen lassen sich beliebig kombinieren bzw. verketten. Hierbei werden die Formatoptionen durch ein Komma getrennt, `{%lfa,%ls ,%lc}` (Erzeugt eine Liste deren Elemente mit „`a`" beginnen und sortiert die Elemente alphabetisch und stellt die Summe aller Elemente der Liste voran). Beachten Sie hierbei bitte, dass die Optionen der Reihe nach angewandt werden. So sollte natürlich die Filterung erfolgen, bevor der Count ausgegeben wird.

Anwendung der verschiedenen Optionen:

Erzwingen von Stellen:

%3m erzwingt mindestens 3 Stellen und stellt `BLANKs` (Leerzeichen) voran

Erzwingen und Auffüllen von führenden Nullen:

%04s erzwingt mindestens 4 Stellen und füllt mit führenden Nullen auf

Gezieltes Ausgeben von Listen (möglich bei allen Befehlen, die Listen zurückgeben):

%l{Index},{Länge} gibt ab `{Index}` `{Länge}` Listenelemente zurück. Wenn `{Länge}` negativ ist, wird rückwärts gezählt, also die Ausgabe herumgedreht.

%lc Fügt der Listenausgabe die Anzahl der beinhalteten Elemente im Format „`...=({Anzahl})|..`“ an

%ls Sortiert die Listenausgabe alphabetisch

%lf{Suchstring} Filtert die Listenausgabe nach dem Inhalt von `{Suchstring}`. Hierbei erfolgt die Suche von links nach rechts.

%lu Ignoriert Elemente der Liste, die mehrfach ausgegeben werden.

Universelle Format-String:

%c{Index},{Länge} gibt bestimmte Zeichen eines Strings aus.

%r{String A},{String B} ersetzt oder löscht die Zeichen aus `{String A}` durch `{String B}`. Ist `{String B}` leer bzw. wird kein Argument angegeben, werden alle Zeichen aus `{String B}` gelöscht. Beide Formatoptionen dürfen an beliebiger Stelle stehen.

Formatierbefehl für xml-Content:

Dieser Formatierer wird verwendet, um Daten aus einem gültigen(!) xml-String zu filtern. Mehrfache Treffer sind möglich und werden mit “|” in der Ausgabe getrennt, sodass weitere Listenformatierer angewendet werden können.

Syntax: `%fxml{NodeName}[={NodeValue}][~{AttributeName}][={AttributeValue}]`

Beispiele:

{%fxmlartist} gibt alle Inhalte der xml Nodes mit dem Namen “artist” aus.

{%fxmlartist=Depeche Mode} möglich aber ohne Auswirkung, da kein Match definiert ist.

{%fxmlartist=Depeche Mode~id} gibt die Werte der Attribute `"id"` aus, wenn der Inhalt der xml Node mit dem Namen `"artist"` `"Depeche Mode"` ist.

{%fxmlartist~id=4711} gibt alle Inhalte der xml Nodes mit dem Namen `"artist"` aus, wenn der Wert des `"id"` -Attributes `"4711"` ist.

{%fxmlartist~id} wie `{%fxmlartist}`: bei zwei fehlenden Kriterien ist Inhalt der Nodes vorrangig.

Rückgabewerte formatieren (bisher nur für Standard-Kommandos möglich, nicht für Commandserver Replays).

Beispiele für die Verwendung der Formatoptionen:

Der normale `<ITUNES><GETCTIME></ITUNES>` Aufruf liefert im Abspielmodus z.B. `521` Sekunden zurück.

Der Aufruf:	<ITUNES><GETCTIME{%02m:%02s}></ITUNES>

Die Ausgabe:	GETCTIME=08:41|OK

Es werden erst Minuten und Sekunden berechnet. Dann werden die Minuten 2-stellig mit führender Null dargestellt, ein ":" als Trenner eingebaut und anschließend die Sekunden 2-stellig angefügt.

Der Aufruf:	<ITUNES><GETCTIME{%02h:%02m:%02s}></ITUNES>
Die Ausgabe:	GETCTIME=00:08:41|OK

Will man nur führende Nullen für einen Wert erzwingen, trägt man nur den "Sekunden"- Formatierer ein:

Der Aufruf:	<ITUNES><GETCTIME{%07s}></ITUNES>
Die Ausgabe:	GETCTIME=0000521|OK

Sollen Argumente formatiert werden (funktioniert prinzipiell genau wie oben) ist zu beachten, dass der Formatstring an das Argument hinten angefügt wird:

Der Aufruf:	<SYS><SETVOL=87{%06s}></SYS>
Die Ausgabe:	<SETVOL=000087>

Beispiele für die Listenausgabe:

<SYS><GETFOLDERINFOLDER{%l 17,5 }=/Users/mmh/Test></SYS> Liefert 5 Ordnernamen ab dem Index 17 als Liste zurück.

<SYS><GETFILESINFOLDER{%l 12,-5 }=/Users/mmh/Test></SYS> Liefert die Dateinamen von Index 12 bis Index 8, beginnend mit Index 12

<ITUNES><GETALLALBUMS{%lc}></ITUNES> Liefert wie folgt die Namen aller Albums aus der iTunes Datenbank und stellt die Anzahl der Elemente (Treffer) der **Ausgabe** voran:

```
<ITUNES>GETALLALBUMS{%lc}=   (4)   |All The Lost Souls|Café
del Mar: The Ibiza Clubmixes (2008)|Embrya|Lounge Top
55 Deluxe|Pacha - The World's Most Famous Club Sound
2008|OK</ITUNES>
```

<ITUNES><GETIDTITLE{%ls}=1080|1082|1084|1086|1088></ITUNES> Liefert den Titelnamen zu den abgefragten ID's und gibt die Namen sortiert zurück.

<ITUNES><GETALLALBUMS{%lfc}></ITUNES> Liefert eine Liste aller Alben, die mit „c" beginnen. (nicht case sensitiv)

<ITUNES><GETALLALBUMS{%lc,%lu}></ITUNES> Liefert eine Liste aller Alben und filtert doppelt vorhandene Albennamen aus.

Beispiele für universelle Formatstrings:

<ITUNES><GETTITLE></ITUNES> Liefert beispielsweise „Cafe Del Mar“

<ITUNES><GETTITLE{%c5,3}></ITUNES> Gibt den String ab Position 5 in der Länge 3 aus, also „Del“

<ITUNES><GETTITLE{%c5,14}></ITUNES> Liefert den String „`Del Mar * Cafe`“. Gibt den String ab Position 5 in der Länge 14 aus. Da die gewünschte Länge des Strings größer ist als der verbleibende Rest des Strings, wird ” * ” eingefügt und vom Beginn des Strings erneut gelesen, bis die 14 Zeichen erreicht sind.

Wenn der `{Index}` -Wert größer als die Stringlänge ist, wird er automatisch heruntergerechnet. Ein "Hinauslaufen" bei hohen `{Index}` -Werten ist also nicht möglich.

<SQUEEZEBOX><ADDANDPLAY=Die besten Hits{%r ,%20}></SQUEEZEBOX> wandelt den Befehl in `<ADDANDPLAY=Die %20 besten %20 Hits>` um, ersetzt also alle „`BLANKS`“ HTML konform in `%20`.

<ITUNES><PLAYPLAYLIST=CRAPLIST{%rCRAP,MY},1{%+1}></ITUNES> wird zu `<ITUNES><PLAYPLAYLIST=MYLIST,2></ITUNES>`

<SYS><SETVAR=TEMP,[TEMP]{%+1.5}></SYS> Erhöht den Wert der Systemvariable `[TEMP]` um 1.5.

<ITUNES><GETIDTITLE{%tmRemotePlaylist}=518|520|522></ITUNES> Erzeugt die Antwort: `<ITUNES><GETIDTITLE{%tmRemotePlaylist}=Hello|Bad|Erotica|OK </ITUNES>`

Wenn man unter der Berücksichtigung des vorherigen Besispiels nun auf:

GETIDTITLE{%tmRemotePlaylist}=*|OK

matcht, kann man die Antwort eindeutig zuordnen ohne Gefahr zu laufen, daß eine andere Abfrage zur Erzeugung einer neuen Liste herangezogen wird. Beispielsweise können hierüber auch mögliche Matchbedingungen in Event Definitionen allein auf den TAG-Namen formuliert werden, also z.B. auf `{%tmRemotePlaylist}=\*|OK`. Der Initiator (`GETIDTITLE` oder `GETIDALBUM`) der Liste bleibt hierbei völlig unberücksichtigt. So können z.B. Ausgaben (m..Remote) mehrfach verwendet werden.

(„`%t`“ hat also keine echte Funktion, sondern bietet lediglich ein Unterscheidungsmerkmal durch das Trennungszeichen „`|`“).

3.4.2 Konvertieroptionen

Verwendung von Konvertierungsoptionen, die ein Argument umrechnen/umwandeln oder auch konvertieren können. Dies ist z.B. notwendig um Werte in HEX zu wandeln oder Werte gezielt zu berechnen, wie nachfolgende Beispiele verdeutlichen sollen.

Die Option wird an **das Argument**, welches umgerechnet werden soll, in „{}“ eingefasst angehängt.

Folgende Konvertierungsoptionen sind z.Z. verfügbar:

Umwandlungen:

%dec wandelt einen HEX-Wert in die entsprechende dezimale Zahl um

%ascii wandelt eine HEX-Sequenz in ASCII Zeichen um

%hex{Stellen} wandelt eine Dezimalzahl oder eine ASCII Sequenz in eine HEX-Zahl mit `{Stellen}` (nur bei Dezimalzahlen) um. `{Stellen}` ist optional. Max. 8 Stellen möglich

%raw sorgt dafür, dass die bei HEX-Command- und Eventservern normalerweise implizit stattfindende Konvertierung des Matches in eine Dezimalzahl unterdrückt wird. Die Kombination von `{%raw,%ascii}` sorgt z.B. dafür, dass eine Hex-Sequenz als ASCII-String dargestellt wird.

%pre{Text} fügt `{Text}` vor dem String ein.

%suf{Text} hängt {Text} an den String an

Rechnenoperationen:

%-{Wert} subtrahiert `{Wert}` von Argument

%+{Wert} addiert `{Wert}` von Argument

%/{Wert} dividiert `{Wert}` von Argument ist `{Wert} = „0“` erfolgt keine Berechnung

%*{Wert} multipliziert `{Wert}` von Argument

Die Optionen können auch mehrfach angewendet und zusammengefasst werden. Hierbei ist zu beachten, dass die Berechnung der jeweiligen Werte sich immer auf das Ergebnis der vorherigen Option bezieht.

Zur Verdeutlichung:

<SYS><SETVOL=0A{%dec,%+30,%/2}></SYS> würde den hexadezimalen Wert `0A` in einen dezimalen Wert `(10)` umwandeln. Der Wert würde nun um `30` addiert werden `(40)` und wiederum durch 2 dividiert werden. Das Ergebnis `20` würde dann gesendet werden.

Anwendung der verschiedenen Optionen:

Beispiele:

<SYS><SETVOL=2F{%dec}></SYS>wird zu <SYS><SETVOL=47></SYS> Wandelt den hexadezimalen Wert „2F“ in die dezimale Zahl 47 um.

<SYS><SAY=414243{%ascii}></SYS>wird zu <SYS><SAY=ABC></SYS> Wandelt die hexadezimalen Werte in die entsprechenden ASCII Zeichen um

<SYS><SAY=15{%hex}></SYS>wird zu <SYS><SAY=F></SYS> Wandelt die dezimale Zahl in eine hexadezimale Zahl um

<SYS><SAY=15{%hex 4 }></SYS>wird zu <SYS><SAY=000F></SYS> Wandelt die dezimale Zahl in eine hexadezimale Zahl um und erzwingt eine Ausgabe mit 4 Stellen.

<SYS><SAY=A1BC{%hex}></SYS>wird zu <SYS><SAY=41314243></SYS> nomos erkennt, dass das Argument nicht nur aus Ziffern besteht und wandelt daher alles in ASCII Zeichencodes um. In diesem Modus wird immer der komplette String umgewandelt. `{Stellen}` hat hier keinen Einfluss.

<SYS><VOLSET=10{%+10}></SYS>wird zu <SYS><VOLSET=20></SYS> Addiert den Wert 10 zu dem Argument

<SYS><VOLSET=10{%\2}></SYS>wird zu <SYS><VOLSET=5></SYS> Dividiert das Argument durch den Wert 2

Komplex: <SYS><SETVOL=0A{%dec,%+30,%*2,%/4,%-3}></SYS>
wird zu <SYS><VOLSET=17> </SYS>

`0A` wird in die dezimale Zahl `10` gewandelt. Dieser Wert wird um `30` addiert. Das Ergebnis wird mit `2` multipliziert. Dieses Ergebnis wiederum durch `4` dividiert und das erneute Ergebnis mit `3` subtrahiert.

3.4.3 Parser Optionen (Match Bedingungen)

Das nomos system generiert diverse Ausgaben auf die verschiedenen Ports. Diese Ausgaben können automatisch, oder aber z.B. über `MAPPINGS` Einträge in den `CommandServer` Definitionen generiert werden. Matches sind case-insensitive.

Um nun diesen Textinhalten gezielt Informationen entnehmen zu können, werden folgende Match- bedingungen / Parser-Optionen verwendet:

Erklärung der Zeichen:

\^ (\#) Matcht ein Zeichen. Je „`^`“ wird jeweils ein Zeichen erwartet. „`#`“ ist hierbei ebenfalls gültig, sollte aber vermieden werden, da auch als Argument verwendet.

***** Matcht eine Zeichenkette

\? **i** gneriert ein Zeichen

\% ignoriert beliebig viele Zeichen

\!{Anzahl}! überspringt `{Anzahl}` Stellen. `{Anzahl}` darf maximal 4 Ziffern haben und muss dezimal angegeben werden. Zu beachten: Bei einem HEX-Match entspricht ein Byte zwei Stellen.

Matches sind case-insensitive. Die weiteren Erläuterungen und Beispiele für die Verwendung finden Sie in den Kapiteln CommandServer und EventServer .

3.4.4 Argumente (Übergabe)

Zeichenketten oder Werte, die nach vorher genannten Matchbedingungen ausgewertet wurden, werden „gepuffert“ und können mit folgenden Argumenten übergeben werden

Erklärung der Zeichen:

\# Übergibt den Inhalt der Matchsequenz (bzw. den „Puffer“)

\$ Übergibt den Inhalt der Matchsequenz (bzw. den „Puffer“ als Liste) ergänzend zu `\$:`

\& Listenindex nur in Verbindung mit „`\$`“

\§ Übergibt die JOIN Nummer (nur `mremote.csv`)

Die weiteren Erläuterungen und Beispiele für die Verwendung finden Sie in den Kapiteln CommandServer und EventServer .

3.4.5 Automatisch generierte Ausgaben (Statusport)

Das nomos system generiert automatisch Broadcast Ausgaben auf den voreingestellten `Port 1038`. So werden automatisch Zustände bzw., Informationen einzelner Applikationen wie z.B. die aktuelle Lautstärke oder aber der aktuell spielende iTunes Titel generiert. Diese Ausgaben können zB über den Eventserver oder innerhalb der mremote Schnittstelle ausgewertet und weitere Aktionen angetriggert werden.

Folgende Daten werden über den Broadcast Port automatisch erzeugt:

ITUNES:

```
<ITUNES>GETPLAYERSTATE=PLAYING|OK|GETPLAYLIST=Musik|OK|
GETTITLE=Let's Get Rocked|OK|GETSTREAMTITLE=|OK|GETARTIST=Def
Leppard|OK|GETALBUM=Adrenalize|OK|GETID=12378|OK|GETINDEX=8|OK|
GETRATING=0|OK</ITUNES>
```

GETPLAYERSTATE= PLAYING/STOPPED/PAUSED

GETPLAYLIST= Name der aktuellen Playliste

GETTITLE= Name des Titels

GETSTREAMTITLE= Name des streaming Titels (IP Radiosender)

GETARTIST= Name des Artisten

GETALBUM= Name des Albums

GETID= Titel ID

GETINDEX= Index des Titels in der aktuellen Playliste

GETRATING= Höhe des Ratings (0-5)

[REMOTE]:

```
<MBA>GETINFO=PLAYING|SHUFFLE=OFF|REPEAT=OFF|TITLE=The A Team|
ARTIST=Ed Sheeran|ALBUM=The A Team - EP|GENRE=Singer/Songwriter|OK</
MBA>
```

GETINFO= PLAYING/STOPPED/PAUSED

SHUFFLE= Shuffle Status

REPEAT= Repeat Status

ITLE= Name des Artisten

ALBUM= Name des Albums

GENRE= Titel ID

DVD:

```
<DVD>GETDVDTITLE=Drums & More Drums|OK</DVD><DVD>GETDVDSTATE=PLAYING|OK|GETTITLE=1|OK|
GETCHAPTER=1|OK</DVD>
<DVD>GETVOL=35|OK</DVD>
```

GETDVDSTATE= PLAYING"/"STOPPED"/"PAUSED

GETTITLE= Name des Titels

GETCHAPTER= Nummer des aktuellen Kapitels

GETVOL= Lautstärke

TV:

```
<TV>GETCH=13|OK|GETCHNAME=3sat|OK|GETPROGINFO=14:15|15:00|Fu Long & Fu Hu - Die Pandas
aus Wien|N/A|15:00|15:45|Almendro - Baum des Lebens|N/A|OK</TV>
<TV>GETVOL=54|OK</TV>
```

GETCH= Aktuelle Kanalnunmmer

GETCHNAME= Aktueller Name des Senders

GETPROGINFO= mit folgendem Inhalt:
Startzeit aktuelle Sendung
Endzeit aktuelle Sendung
Kurzinfo aktuelle Sendung Langtext aktuelle Sendung
Startzeit nächste Sendung
Endzeit nächste Sendung
Kurzinfo nächste Sendung
Langtext nächste Sendung

GETVOL= Applikationslautstärke

GETTUNER= Name des Tuners, der aktuell im Vordergrund (nur bei mehreren vorhanden Tunern im System)

SYS:

```
<SYS>GETVOL=INT|VOL=68|MUTE=OFF|OK</SYS>
<SYS>GETSS=ON|OK</SYS>
```

GETVOL= mit folgendem Inhalt:
Audioausgabe `EXT/INT`
Lautstärke `VOL={0-100}`
Mute `MUTE={ON/OFF}`

GETSS= Status des Bildschirmschoners `{ON/OFF}`

Follow-Me-Status:

„FMINFO=...|OK“ Übergabe der Aufrufparameter für das Abspielen einer DVD auf einem entfernten System

Folgende Klassen erzeugen ebenfalls Broadcast Statusausgaben. Die zugehörigen Erklärungen finden Sie in den jeweiligen Kapiteln.

xPL (wenn gültige (= in der `.csv` definierte) xpl-message empfangen wurde)

HS (immer, wenn etwas empfangen wurde)

KNX (immer, wenn die Adresse in der `.esf` definiert ist)

BAOS (wenn der Datenpunkt in der `.csv` definiert wurde)

ZWAVE (Statusänderungen angemeldeter Geräte)

HUE (Statusänderungen angemeldeter Geräte)

SONOS (Titelinformationen und Status Meldungen)

{CommandServer} (wie unter `MAPPINGS` definiert)

3.5 Befehlsklassen

Das nomos system verfügt über folgende interne Befehlsklassen:

Klasse	Beschreibung	os x	linux
BROWSER	Steuerung des integrierten Internet-Browsers (Safari)	x	
DVD	Steuerung der Applikation zur DVD-Wiedergabe	x	
ITUNES	Steuerung der Multimedia-Applikation iTunes	x	
SLIDE	Steuerung der integrierten Picture-Slideshow	x	
SCRIPT	Befehlsklasse zur Verwaltung von lokalen Scripten	x	x
SCRIPTLOAD	Befehlsklasse zum Laden von Scripten auf das nomos system	x	x
SYS	Applikationsunabhängige Systembefehle	x	x(teil)
TV	Steuerung der TV-Applikation (EyeTV)	x	
VLC	Steuerung des VideoLAN-Clients	x	
UDP	Senden „freier“ UDP Datagramme	x	x
TCP	Senden „freier“ TCP Pakete	x	x
WINDOW	Befehlsklasse zur Anzeige von freien Fensterinhalten (Text/Grafik/Browser)	x	
KNX	KNX/IP - Support	x	x
BAOS	BAOS Objektserver - Support	x	x
HS	GIRA Homeserver KO-Gateway Support	x	x
TIMER	Timer Support	x	x
QUICKTIME	Steuerung der Multimedia-Applikation Quicktime	x	
[dynamisch]	Steuerung von Remote und Sonos Klassen	x	x

Die Einschränkungen des LINUX Befehlssatzes beziehen sich nur auf die lokale Steuerung der Software- Applikationen, wie sie nur auf dem Apple OS X System verfügbar sind. Die Befehle können jedoch benutzt werden, um diese Funktionen auf entfernte Systeme auszuführen sofern dort die nomos Software verfügbar ist. Es kann also ein zB iTunes oder Safari Browser über eine nomos Box auf einen entfernten Apple Computer angesteuert werden. Siehe hierzu auch die *Relay Funktion*.

3.6 Standard Befehlssatz

Der Standard Befehlssatz ist fest in dem nomos system eingebunden. Der Befehlssatz ist mit wenigen Ausnahmen über alle Plattformen hinweg kompatibel. Der Befehlssatz kann über die CommandServer Schnittstelle nahezu beliebig erweitert werden.

3.6.1 BROWSER

Befehlssatz zur Steuerung der Safari Web Browser Applikation (nur OS X).

Kommando	Argument	Beschreibung
ON	-	startet den Browser
OFF	-	beendet den Browser
CLOSEAWIN	-	schließt alle Browserfenster
CLOSEFWIN	-	schließt das letzte Browserfenster
MINFWIN=	{0} oder {1}	minimiert (1) bzw. maximiert (0) das letzte Fenster
MINAWIN=	{0} oder {1}	wie MINFWIN, für alle Fenster
URL=	{URL}	öffnet die angegebene URL in einem neuen Fenster
ZOOM=	{0} oder {1}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax, ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
WINSIZE=	FULL oder {XMAX, YMAX} oder {XMIN, YMIN,XMAX,YMAX}	Ändert die Fenstergröße (Ursprung unten links)
Erklärung: `FULL`: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt) `XMAX,YMAX`: zentriert das Fenster mit der Ausdehnung Xmax,Ymax auf dem Bildschirm `XMIN,YMIN,XMAX,YMAX`: positioniert das Fenster mit der Ausdehnung Xmax,Ymax und dem Ursprung Xmin,Ymin auf dem Bildschirm		

3.6.2 DVD

Befehlssatz zur Steuerung der DVD-Player Applikation (nur OS X).

Kommando	Argument	Beschreibung
ON	-	startet die DVD Player Applikation
OFF	-	beendet die DVD Player Applikation
PLAY	-	spielt die eingelegte DVD ab
STOP	-	stoppt die DVD
PAUSE	-	pausiert die DVD
FF	-	schneller Vorlauf
REW	-	schneller Rücklauf
EJECT	-	wirft die eingelegte DVD aus
URL=	{URL}	öffnet die angegebene URL
FSON	-	aktiviert den Vollbildmodus
FSOFF	-	deaktiviert den Vollbildmodus
MUTEON	-	schaltet den Ton aus
MUTEOFF	-	schaltet den Ton ein
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
UPKEY	-	simuliert einen Tastendruck (Menü-Navigation)
DNKEY	-	simuliert einen Tastendruck (Menü-Navigation)
LEKEY	-	simuliert einen Tastendruck (Menü-Navigation)
RIKEY	-	simuliert einen Tastendruck (Menü-Navigation)
ENKEY	-	simuliert einen Tastendruck (Menü-Navigation)
GETPTIME	-	liefert die aktuelle Spielzeit in Sekunden
SETPTIME=	{1...x}	springt zur angegebenen Spielzeit
GETCTIME	-	liefert die gesamte Spielzeit in Sekunden
GETCHAPTER	-	liefert das aktuelle Kapitel
SETCHAPTER=	{1...x}	springt zum angegebenen Kapitel
GETALLCHAPTERS	-	liefert die Gesamtzahl aller Kapitel
GETMEDIASTATE	-	liefert den Medien-Status der DVD
GETMENUSTATE	-	liefert den DVD-Menustatus
GETDVDSTATE	-	liefert den Status der DVD
GETCONVIS	-	liefert den Status des Controller-Fensters
SETCONVIS=	{0} oder {1}	(de-)aktiviert das Controller-Fenster
SETCONPOS=	{1...x},{1...y}	setzt die Position des Controller-Fensters
GETINFVIS	-	liefert den Status des Info-Fensters
SETINFVIS=	{0} oder {1}	(de-)aktiviert das Info-Fenster
SETINFPOS=	{1...x},{1...y}	setzt die Position des Info-Fensters
GETTITLE		liest die aktuelle Titelnummer der eingelegten DVD aus
SETTITLE=	{1...x}	springt zur angegebenen Titelnummer
GETDVDTITLE		liefert den aktuellen DVD-Titel
SETVOL=	{0...100}	ändert die Lautstärke des DVD-Players
GETVOL		liefert die Lautstärke der DVD-Applikation im Bereich 0...100
WINSIZE=	FULL oder {XMAX, YMAX} oder{XMIN, YMIN,XMAX,YMAX}	FULL: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt)

Kommando	Argument	Beschreibung
`XMAX,YMAX`: zentriert das Fenster mit der Ausdehnung Xmax,Ymax auf dem Bildschirm `XMIN,YMIN,XMAX,YMAX`: positioniert das Fenster mit der Ausdehnung Xmax,Ymax und dem Ursprung Xmin,Ymin auf dem Bildschirm		

3.6.3 ITUNES

Befehlssatz zur Steuerung der iTunes Applikation (nur OS X).

Kommando	Argument	Beschreibung
ADD#	{Pfad}	fuegt eine Datei zur aktuellen Playliste hinzu
ADDIDTOPLAYLIST#	{ID},{NAME}	fuegt Titel {ID} zur Playlist {NAME} hinzu
CREATEPLAYLIST#	{NAME}	erzeugt eine neue Playlist {NAME}
CREATEPLAYLIST FROM-LASTSEARCH#	{PL-Name} wird kein Name angegeben, wird die Liste autom. benannt	erzeugt eine Playliste {PL-Name} mit dem Inhalt des Ergebnisses der letzten Suchabfrage. {PL-Name} ist optional.
DELETEPLAYLIST#	{NAME}	loescht die Playlist {NAME}
DISABLECF		Beendet das iTunes Cover Flow
ENABLECF		Startet das iTunes Cover Flow im Vollbildmodus
FF		schneller Vorlauf
FRONT		holt die Applikation nach vorne
GETALBUM		liefert das Album des aktuellen Titels
GETALLALBUMS		liefert eine Liste aller Alben zurueck (max. 250)
GETALLARTISTS		liefert eine Liste aller Artist zurueck (max. 250)
GETALLIDSOF PLAYLIST#	{Playlist}	liefert alle IDs in {Playlist} (max. 250)
GETALLPLAYLISTS		liefert eine Liste aller vorhanden Playlisten
GETALLTITLES		liefert eine Liste aller Titel zurueck (max. 250)
GETARTIST		liefert den Kuenstler des aktuellen Titels
GETCTIME		liefert die gesamte Spielzeit in Sekunden
GETID		liefert die {ID} des aktuellen Titels
GETIDALBUM#	{ID}	liefert das Album zur angegebenen {ID}
GETIDARTIST#	{ID}	liefert den Interpreten zur angegebenen {ID}
GETIDRATING#	{ID}	liefert die Userbewertung der {ID} Tracks im Bereich (0-5) synchron der Anzahl Sterne
GETIDTITLE#	{ID}	liefert den Titel zur angegebenen {ID}
GETINDEX		liefert den Index des aktuellen Titels in der aktuellen Playliste
GETMUTE		liefert den Mute-Status
GETPLAYERSTATE		liefert den Status des Players
GETPLAYLIST		liefert die aktuelle Playliste
GETPLAYLISTCOUNT		liefert die Anzahl der vorhandenen Playlisten (max. 250)
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
GETRATING		liefert die Userbewertung des aktuellen Tracks im Bereich (0-5) synchron der Anzahl Sterne
GETREPEAT		gibt den aktuellen Repeatmodus der Playliste zurueck
GETSHUFFLE		gibt den aktuellen Playmodus der Playliste zurueck
GETSTREAMTITLE		liefert den Titel des aktuellen Streams
GETTITLE		liefert den aktuellen Titel
GETVISSTATE		liefert den Visualisierungs-Status
GETVOL		liefert die aktuelle Applikationslautstaerke
HIDE		versteckt die Applikation
HIDEPLAYER		blendet das iTunes-Hauptfenster aus
MUTEOFF		schaltet den Ton aus

Kommando	Argument	Beschreibung
MUTEON		schaltet den Ton ein
NEXT		springt zum naechsten Titel
OFF		beendet die iTunes-Applikation
ON		startet die iTunes-Applikation
PAUSE		pausiert den aktuellen Titel
PLAY		spielt den aktuellen Titel ab
PLAYF#	{Pfad}	spielt die in {Pfad} angegebene Datei ab
PLAYID#	{ID}	spielt den Titel `{ID}` ab
PLAYPLAYLIST#	{Name},{Index}	spielt die Playliste `{Name}` ab. Wird zus. `{Index}` angegeben, beginnt die Wiedergabe ab diesem Titel in der Playliste. Siehe auch `GETINDEX`.
PREV		springt zum vorigen Titel
RESUME		setzt die Wiedergabe fort
REW		schneller Ruecklauf
SEARCHALBUM#	{Album}	liefert alle IDs, die zu `{Album}` gehoeren
SEARCHARTIST#	{Suchstring}	liefert eine Liste der Interpreten zurueck, auf die der Suchstring passt (Limit: 50)
SEARCHTITLE#	{Suchstring}	liefert eine Track-ID -Liste der Titel zurueck, auf die der `{Suchstring}` passt (Limit: 50)
SETIDRATING#	{ID},{0...5}	setzt die Bewertung des iTunes Titels `{ID}`, (0-5) synchron der Anzahl Sterne
SETPTIME#	{1...x}	springt zur angegebenen Spielzeit
SETRATING#	{0...5}	setzt die Bewertung des aktuellen iTunes Titels (0-5) synchron der Anzahl Sterne
SETREPEAT#	{ONE/ALL/OFF}	setzt den Repeatmodus der aktuellen Playlist
SETSHUFFLE#	{ON/OFF}	schaltet den Modus Shuffle der aktuellen Playlist ein o. aus
SETVISSTATE#	„0“ oder „1“	(de-)aktiviert die Visualisierungen
SETVOL#	{0...100}	Setzt die Systemlautstaerke (iTunes) auf 0-100%
SHOW		macht die Applikation sichtbar
SHOWPLAYER		blendet das iTunes-Hauptfenster ein
STOP		stoppt den aktuellen Titel
URL#	{URL}	oeffnet die angegebene URL (Streaming)
WFORMAT#	xmin,ymin,xmax,ymax	skaliert das aktuelle Fenster auf den angeg.. Bereich
WINSIZE#	FULL o. {XMAX,YMAX} o. {XMIN,YMIN,XMAX, YMAX}	FULL oder XMAX,YMAX oder XMIN,YMIN,XMAX,YMAX

3.6.4 SLIDE

Befehlssatz zur Steuerung der Slide Show aus der iPhoto Applikation (nur OS X).

Kommando	Argument	Beschreibung
ON	-	startet die iPhoto-Applikation
OFF	-	beendet die iPhoto-Applikation
START	-	startet die Slideshow
STOP	-	beendet die Slideshow
PAUSE	-	pausiert die Slideshow
RESUME	-	setzt die Slideshow fort
PREV	-	springt zum vorigen Bild
NEXT	-	springt zum nächsten Bild
IMPORT=	{Pfad}	importiert die Bilder im angegebenen Pfad
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
GETALBUMLIST	-	liefert eine Liste aller Alben
SELALBUM=	{Name.suffix}	selektiert das Album `{Name.suffix}`
GETALBUM	-	liefert den Namen des aktuellen Albums

3.6.5 SCRIPT

Befehlssatz für die Bearbeitung von Scripten

Kommando	Argument	Beschreibung
RUN=	{Name.suffix}	führt das `.myh` -Script `{Name.suffix}` aus
ARG=	{Wert}	Übergibt einen oder mehrere variablen Werte an das Script. (füllt sequentiell den o. die Platzhalter `ARG`)
RUNAS=	{Name.suffix}	führt das Applescript `{Name.suffix}` aus
RUNSS=	{Name.suffix}	führt das Shell-Script `{Name.suffix}` aus
DELETE=	{Name.suffix}	löscht das Script `{Name.suffix}`
RENAME=	{Name.suffix} ={Name_neu.suffix}	benennt `.myh` Script `{Name.suffix}` in `{Name_neu.suffix}` um
LIST	-	listet alle `.myh` -Scripte im scripts-Ordner
LISTAS	-	listet alle Applescripte im scripts-Ordner
LISTSS	-	listet alle Shell-Scripte im scripts-Ordner
SHOW=	{Name.suffix}	zeigt den Inhalt des Scripts `{Name}`

3.6.6 SCRIPTLOAD

Erweiterter Befehlssatz für die Bearbeitung von Scripten. Diese Kommandos werden in der Regel von Autorensoftware bzw. Editoren verwendet.

Kommando	Argument	Beschreibung
NAME=	{Name.suffix}	legt ein neues Script `{Name.myh}` an, alle nachfolgenden Klassen und Argumente definieren den Inhalt des `.myh` Scripts, bis der Klassen-Terminator `</SCRIPTLOAD>` gelesen wurde

3.6.7 SYS

Kommando	Argument	OSX only	Beschreibung
SETVOL=	{0...100}	x	Setzt die Systemlautstärke auf 0-100%
VOLUP=	{0...100}	x	erhöht die Systemlautstärke um 0-100%
VOLDN=	{0...100}	x	verringert die Systemlautstärke um 0-100%
MUTEON	-	x	schaltet den Systemton ab
MUTEOFF	-	x	schaltet den Systemton ein
GETVOL	-	x	iefert die Systemlautstärke
GETSTATE	-	x	liefert Informationen über den Applikations-Status
UPKEY	-	x	simuliert den Tastendruck „Cursor hoch“
DNKEY	-	x	simuliert den Tastendruck „Cursor runter“
LEKEY	-	x	simuliert den Tastendruck „Cursor links“
RIKEY	-	x	simuliert den Tastendruck „Cursor rechts“
ENKEY	-	x	simuliert den Tastendruck „Enter“
SPCKEY	-	x	simuliert den Tastendruck „Leertaste“
TABKEY	-	x	simuliert den Tastendruck „Tabulator“
PUSHUPKEY		x	simuliert den “hold” Tastendruck „Cursor hoch“
PUSHDNKEY		x	simuliert den “hold” Tastendruck „Cursor runter“
PUSHLEKEY		x	simuliert den “hold” Tastendruck „Cursor links“
PUSHRIKEY		x	simuliert den “hold” Tastendruck „Cursor rechts“
SETMPOS=	{0...x},{0...y}	x	setzt den Mauszeiger auf x,y
MCLICK=	{0...x},{0...y} oder HERE	x	{x,y} oder HERE - simuliert einen Mausklick an der Position x,y, ohne den Mauszeiger zu bewegen. HERE klickt an der aktuellen Position des Mauszeigers
GETMPOS			liefert die aktuellen Koordinaten des Mauszeigers
ENABLEFR	-	x	aktiviert die FrontRow-Applikation (ab OSX 10.7 nicht mehr verfügbar)
DISABLEFR	-	x	deaktiviert die FrontRow-Applikation (ab OSX 10.7 nicht mehr verfügbar)
ENABLESS	-	x	aktiviert den Bildschirmschoner
DISABLESS	-	x	deaktiviert den Bildschirmschoner
ENABLEDB	-	x	aktiviert das Dashboard
DISABLEDB	-	x	deaktiviert das Dashboard
SHOWDOCK	-	x	zeigt das Dock
HIDEDOCK	-	x	versteckt das Dock
HIDEALL	-	x	Bringt alle Anwendungen in den Hintergrund (Hide)
HIDESTOPFRONT	-	x	versteckt, stoppt und mutet die oberste Applikation
OPEN=	{Name.suffix}		versucht, die Datei {Name.suffix} zu öffnen
SHUTDOWN	-		fährt das System herunter
RESTART	-		startet das System neu
EJECT	-	x	wirft die CD/DVD im Laufwerk aus
SLEEP	-	x	versetzt das System in den Ruhezustand
HELLO	-		liefert Informationen über den nomos
GETLICENSEINFO	-		liefert Informationen über die aktuell installierte Lizenz. Format der Antwort: `{Gültigkeit}|{Commandserver}|{Event-Support}|{xPL-Support}|{BAOS-Support}|{Extended-Buttons-Support}`

Kommando	Argument	OSX only	Beschreibung
GETSYSTEMLIST			liefert die Information aller vorhandenen nomos und nomos oem Systeme
FADE=	{1} oder {0}	x	blendet den Bildschirm aus bzw. ein.
MESSAGE=	{„text“}	x	Blendet den Bildschirm aus und zeigt den Text {text} zentriert auf dem Bildschirm an. Ein „|“ im Text-String erzeugt einen Zeilenvorschub. Entfällt in späteren Versionen.
MESSAGEOFF=		x	Blendet den mit MESSAGE angezeigten Text aus und zeigt den Desktop an. Entfällt in späteren Versionen.
SAY=	{Text}	x	Übergibt einen Text an die Sprachausgabe
RESTARTSERVICE			Startet und resettet den Daemon neu.
CMDKEY=#	{Keycode}	x	Simuliert Tastendruck [Apfel]+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
ALTCMDKEY=	{Keycode}	x	Simuliert Tastendruck [ALT]+[Apfel]+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
CTRLCMDKEY#	{Keycode}	x	Simuliert Tastendruck [CTRL]+[Apfel]+{Keycode}. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
SHIFTCMDKEY#	{Keycode}	x	Simuliert Tastendruck [SHIFT]+[CMD]+{Keycode}
KEY#	{Keycode}	x	Simuliert Tastendruck. Wichtig: Es wird der Apple- Tastaturcode erwartet, nicht der ASCII-Code! Eine entsprechende Tabelle finden Sie im Anhang der Dokumentation
FMINIT	-	x	Initialisiert den Follow-me Status
FMREAD	-	x	liefert den generierten Follow-me String. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMEXIT	-	x	verlässt den Follow-me Status. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMABORT	-	x	bricht den Follow-me ab und setzt die Wiedergabe fort. (Setzt ein erfolgreiches vorangegangenes FMINIT voraus.)
FMINFO	-	x	liefert den aktuellen Follow-me Status (TV,DVD,...) (auch im Broadcast vorhanden)
GETSRVLIST	-		interner Befehl für das Einleiten zum Auslesen der AddOn Befehle. Nach Auslösen wird das Menu um evtl. vorhandene AddOn’s erweitert
GETXPLLIST	-		interner Befehl für das Einleiten zum Auslesen der xPL Befehle. Nach Auslösen wird das Menu um evtl. vorhandene AddOn’s erweitert
GETFILENUMIN FOLDER=	{Pfad}		liefert die Zahl der in {Pfad} vorhandenen Dateien
GETFOLDERNUMIN FOLDER=	{Pfad}		liefert die Zahl der in {Pfad} vorhandenen Verzeichnisse
GETFILESINFOLDER=	{Pfad,Index=x, Length=y}		liefert eine Liste der in {Pfad} vorhandenen Dateien, Index und Length sind optional
GETFOLDERSIN FOLDER=	{Pfad,Index=x, Length=y}		liefert eine Liste der in {Pfad} vorhandenen Verzeichnisse, Index und Length sind optional

Kommando	Argument	OSX only	Beschreibung
OPENBUTTONS=	{ButtonLeisten Name}	x	öffnet eine Button Leiste
CLOSEBUTTONS=	{ButtonLeisten Name}	x	Schließt eine Button Leiste
SETBUTTONPUSHED=	{ButtonLeisten Name}	x	setzt den Zustand eines Buttons auf betätigt (wenn Toggle Button)
SETBUTTON RELEASED=	{ButtonLeisten Name}	x	setzt den Zustand eines Buttons auf nicht betätigt (wenn Toggle Button)
CLOSECURRENT BUTTONS		x	schließt die aktuell geöffnete Buttonleiste
SETBUTTONLABEL=	{ButtonLeisten Name}, Index{1-10}, {LabelText}	x	Verändert flüchtig das Label des Buttons
SETALTBUTTON LABEL=	{ButtonLeisten Name}, Index{1-10}, {LabelText}	x	Verändert flüchtig das 2. Label eines `SWITCH` Buttons
SHOWKEYBOARD		x	Blendet die Eingabehilfe ein (Keyboard)
HIDEKEYBOARD		x	Blendet die Eingabehilfe aus (Keyboard)
GETVAR=	{Variablenname}		liest den Wert einer System-Variablen
SETVAR=	{Variablenname}, {Wert}		ändert den Wert einer System-Variablen bzw. legt sie an
DELVAR=	{Variablenname}		löscht eine System-Variable
GETALLVARS			Abfrage und Ausgabe der Werte aller vorhandenen Variablen
LOCKVARS=	{ON/OFF}		Verriegelt oder ermöglicht das Schreiben der Variablen auf den Datenträger
WAITVAR=	{Variablenname}, {Wert}{,{Timeout}}		Hält die aktuelle Sequenz so lange an, bis die Sysvar `{Variablenname}` den Wert `{Wert}` angenommen hat. Wird zusätzlich `{Timeout}` (in ms) angegeben, so wird `{Timeout}` - Millisekunden gewartet. Wenn die Zeit abgelaufen ist und `{Variable}` niemals `{Wert}` hatte, wird die restliche Sequenz NICHT abgearbeitet, sondern ignoriert. Hiermit lässt sich ein zeitabhängiges "`IF -> THEN`" nachbilden.
GETSERIALPORTLIST			listet alle im System vorhandene, serielle Ports auf
GETSPEAKERLIST		x	listet alle verfügbaren Airports mit Namen auf ("Computer" ist der lokale Ton des Apple Computers) - Airfoil
GETSPEAKERSOURCE		x	gibt die aktuelle Applikation bzw. den aktiven Eingang aus, dessen Ton gerade gestreamt wird - Airfoil
SETSPEAKER SOURCE=	{Appname}	x	wählt die Applikation aus, die gestreamt werden soll - Airfoil
SETSPEAKERSOURCE DEVICE=	{Eingang}	x	wählt den Audio-Eingang aus, der gestreamt werden soll - Airfoil
GETAUDIODEVICE LIST		x	listet alle verfügbaren Audio-Eingänge auf - Airfoil
GETSPEAKERVOL=	{Airport-Name}	x	Abfrage der Lautstärke des gewählten Airport -Airfoil
SETSPEAKERVOL=	{Airport-Name}, {0...100}	x	setzt die Lautstärke des gewählten Airports absolut, wird der Name weggelassen, wird die Lautstärke für alle Airports (inkl. "Computer") gesetzt - Airfoil

Kommando	Argument	OSX only	Beschreibung
SPEAKERVOLUP=	{Airport-Name}, {0...100}	x	erhöht die Lautstärke um den angegebenen Wert, der Name ist auch hier optional - Airfoil
SPEAKERVOLDN=	{Airport-Name}, {0...100}	x	senkt die Lautstärke, sonst wie `SPEAKERVOLUP` - Airfoil
ENABLESPEAKER=	{Airport-Name}	x	schaltet den gewählten Airport ein - Airfoil
DISABLESPEAKER=	{Airport-Name}	x	schaltet den gewählten Airport aus - Airfoil
ENABLEALL SPEAKERS		x	schaltet alle Airports ein - Airfoil
DISABLEALL SPEAKERS		x	schaltet alle Airports aus - Airfoil
GETSPEAKERSTATE=	{Airport-Name}	x	liefert den Status des gewählten Airports: `ENABLED / DISABLED / NOT_AVAILABLE` - Airfoil
SETDIGITALJOIN=	{Nummer},{Wert}		mRemote Support - setze digital `Join (0/1)`
SETANALOGJOIN=	{Nummer},{Wert}		mRemote Support - setze analog `Join (0-65535)`
SETSERIALJOIN=	{Nummer},{Wert}		mRemote Support - setze serial `Join (String)`
WAKEONLAN=	{MAC-Adresse}		sendet ein „Magic Packet“ für die angegebene MAC-Adresse. Die Schreibweise der Mac-Adresse ist dabei egal, solange exakt `6 Octets` angegeben werden.
GETAUDIOOUTPUT LIST		x	listet alle verfügbaren Audio-Ausgabegeräte mit Namen auf
GETAUDIOOUTPUT		x	liefert das aktuelle Ausgabegerät
SETAUDIOOUTPUT=	{Name}	x	schaltet auf Ausgabegerät `{Name}` um
SETDISPLAYMODE=	{x,y,bpp},{x,x}, “0” oder “NORMAL”	x	Ändert die Bildschirmauflösung, wenn unterstützt `{x,y,bpp}, {x,x}, "0"` oder `"NORMAL"`
GETTIME			liefert die aktuelle Systemzeit im KNX `EIS3` Format
GETDATE			liefert das aktuelle Systemdatum im KNX `EIS3` Format
SETCOUNTER=	{Name}{,Startwert}{,Schrittweite}		initialisiert den Counter `{Name}`. `{Startwert}` und `{Schrittweite}` sind optional. Default ist `0` für `{Startwert}` und `1` für `{Schrittweite}`
SETCOUNTERSTEP=	{Name}{,Schrittweite}		ändert die Schrittweite eines bestehenden Counters `{Name}`, `0` für `{Schrittweite}` hält den Counter an
GETCOUNTER=	{Name}		liefert den aktuellen Wert des Counters `{Name}`, ohne sie zu verändern
GETALLCOUNTERS			liefert die aktuellen Werte aller Counter, ohne diese zu verändern
GETREMOTEDEVICE LIST			liefert eine Liste aller in der remote.csv definierten Geräte
GETVPNIP			liefert die VPN-IP, falls eine VPN Verbindung besteht
REMOTEPAIRING=	{ON/OFF}		Startet `{ON}` /Beendet `{OFF}` den Pairing Mode für iTunes und AppleTV in der remote Klasse
CONVERTHSVTORGB#	{H(0-360),S(0-255), V(0-255)}		Konvertiert Farbwerte aus dem HSV-Farbraum in RGB. Werte: H= Farbwinkel (0-360°) - S= Saettigung (0-255) - V= Helligkeit (0-255), Rueckgabe = R,G,B im Wertebereich 0-255

Kommando	Argument	OSX only	Beschreibung
CONVERTRGB-TOHSV#	{R(0-255)}, {G(0-255)}, {B(0-255)}		Konvertiert RGB Farbwerte in den HSV-Farbraum. Werte: R=(0-255) - G=(0-255) - B=(0-255)
GETSONOSDEVICE-LIST			liefert die Klassennamen aller in der `sonos.csv` definierten Geräte
GETXBMCDEVICE-LIST			liefert die Klassennamen aller in der `xbmc.csv` definierten Geräte

Hinweis zum SETDISPLAYMODE-Befehl:

Wenn die Angaben nicht umsetzbar sind (Auflösung und/oder Farbtiefe), wird automatisch der nächstmögliche Modus eingestellt (siehe Log) Es ist nicht möglich, Modi einzustellen, die vom Bildschirm nicht unterstützt werden. Beim Beenden von mmh wird der Bildschirm auf den ursprünglichen Modus zurückgesetzt.

3.6.8 TV (Elgato eyeTV)

Befehlssatz zur Steuerung der eyeTV Applikation (nur OS X).

Kommando	Argument	Beschreibung
SETVOL=	{0...100}	Setzt die Systemlautstärke (TV) auf 0-100%
ON	-	startet die TV-Applikation
OFF	-	beendet die TV-Applikation
PLAY	-	startet die Wiedergabe
PAUSE	-	pausiert die Wiedergabe
STOP	-	stoppt die Wiedergabe
FF	-	schneller Vorlauf
REW	-	schneller Rücklauf
SLOWF	-	Zeitlupe vorwärts
SLOWR	-	Zeitlupe rückwärts
SKIPF	-	springt nach vorne
SKIPR	-	springt zurück
RECSTART	-	startet die Aufnahme
RECSTOP	-	beendet die Aufnahme
CHUP	-	schaltet einen Kanal nach oben
CHDN	-	schaltet einen Kanal nach unten
FSON	-	aktiviert den Vollbildmodus
FSOFF	-	deaktiviert den Vollbildmodus
ZOOM=	{1} oder {0}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax, ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
HIDECON	-	versteckt das Konsolenfenster
SHOWCON	-	zeigt die Konsolenfenster
CH=	{1...x}	wählt den Kanal
GETCH	-	liefert die aktuelle Kanalnummer
GETCHNAME=	{1...x}	liefert den Kanalnamen zur Kanalnummer
MUTEON	-	schaltet den Applikationston aus
MUTEOFF	-	schaltet den Applikationston ein
ENABLEEGP	-	Schaltet die EPG Ansicht ein
DISABLEEPG	-	Schaltet die EPG Ansicht aus
GETPROGINFO		liefert Informationen über die aktuelle und die nächste Sendung im Format: `{Start}|{Stop}|{Titel der aktuellen Sendung}|{Kurzbeschreibung}|{Start}|{Stop}|{Titel der nächsten Sendung}|{Kurzbeschreibung}`
WINSIZE=	FULL/{XMAX,YMAX}/ {XMIN,YMIN,XMAX, YMAX}	Ändert die Fenstergröße (Ursprung unten links)

Erklärung:

Kommando	Argument	Beschreibung
FULL: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt). XMAX,YMAX: zentriert das Fenster mit der Ausdehnung XMAX, YMAX auf dem Bildschirm. XMIN,YMIN,XMAX,YMAX: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm.		
ENABLEMENU		blendet das eyeTV-OSD-Menu ein
DISABLEMENU		blendet das eyeTV-OSD-Menu aus
ENABLEFSEPG		blendet die Vollbild EPG Ansicht ein
DISABLEFSEPG		blendet die Vollbild EPG Ansicht aus

3.6.9 VLC

Befehlssatz zur Steuerung der VLC Applikation (nur OS X).

Kommando	Argument	Beschreibung
ON	-	aktiviert den VideoLAN-Client
OFF	-	beendet den VideoLAN-Client
URL=	{URL}	öffnet die Datei/den Stream {URL}
TPLAY	-	toggelt den Play-Status
STOP	-	stoppt die Wiedergabe
PREV	-	springt zum vorigen Titel
NEXT	-	springt zum nächsten Titel
TFS	-	toggelt den Vollbild-Modus
TMUTE	-	toggelt den Mute-Status
VOLUP	-	erhöht die Lautstärke
VOLDN	-	reduziert die Lautstärke
ZOOM=	{1} oder {0}	zoomt (1) bzw. reduziert (0) das aktuelle Fenster
WFORMAT=	{xmin,ymin,xmax, ymax}	skaliert das aktuelle Fenster auf den angeg. Bereich
SHOW	-	macht die Applikation sichtbar
HIDE	-	versteckt die Applikation
FRONT	-	holt die Applikation nach vorne
WINSIZE=	FULL/{XMAX,YMAX}/ {XMIN,YMIN,XMAX, YMAX}	Ändert die Fenstergröße (Ursprung unten links)
Erklärung: `FULL`: streckt das Fenster auf Bildschirmauflösung (soweit die Applikation dies zulässt). `XMAX,YMAX`: zentriert das Fenster mit der Ausdehnung XMAX, YMAX auf dem Bildschirm. `XMIN,YMIN,XMAX,YMAX`: positioniert das Fenster mit der Ausdehnung XMAX,YMAX und dem Ursprung XMIN,YMIN auf dem Bildschirm.		

3.6.10 UDP/TCP

Die UDP/TCP Befehlsklasse bietet die Möglichkeit beliebige Protokolle direkt an ein Device zu senden. Bei Verwendung dieser Option wird kein zusätzlicher CommandServer benötigt. Die Klasse <TCP> ist konform zu der Klasse <UDP>.

Kommando	Argument	Beschreibung
IP=	{IP Adresse}x	IP Adresse des Empfängers in Punktnotation
PORT=	{PORT}	Port Adresse des Empfängers
MODE=	{„HEX“,“ASCII“, “BIN“}	legt den Übertragungsmodus fest (z.Z. nur HEX)
TIME=	{t} in Sekunden 0 - 3600	optional, legt den Timeout in Sekunden fest, bis zu dem auf eine Antwort gewartet wird
DATA=	{Daten}	Dateninhalt des UDP-Paketes im MODE-Format

Beispiele:

<UDP><IP=192.168.50.1><PORT=4000><MODE=HEX><TIME=5><DATA=414243></UDP>

Sendet ein UDP-Paket mit dem Inhalt `"ABC"` an die `IP 192.168.50.1` auf `Port 4000` und wartet `5` Sekunden auf Antwort(en). Innerhalb des Timeouts empfangene Daten werden sofort weitergereicht, bis der Timeout abgelaufen ist. Wird kein Timeout angegeben (oder `TIME=0`), wird das Paket nur versendet, aber keine Antwort abgewartet. Der maximal mögliche Timeout Wert ist auf `3600s` festgesetzt.

<UDP><IP=192.168.50.125><PORT=1037><MODE=ASCII><TIME=0><DATA= <SYS><MUTE></SYS> </UDP>

Sendet den System-Befehl `Mute` an das “entfernte” nomos system mit der Adresse `192.168.50.125`

Nach Absetzen eines UDP-Paketes kehrt der Daemon sofort wieder in den Kommandomodus zurück und verarbeitet evtl. weitere anstehende Kommandos.

Die Verwendung von TCP Verbindungen erfolgt ähnlich:

<TCP><IP=192.168.50.1><PORT=4000><MODE=HEX><TIME=5><DATA=414243></TCP>

Sendet ein TCP Paket mit dem Inhalt `"ABC"` an die `IP 192.168.50.1` auf `Port 4000` und wartet `5` Sekunden auf Antwort(en). Innerhalb des Timeouts empfangene Daten werden sofort weitergereicht, bis der Timeout abgelaufen ist. Wird kein Timeout angegeben `(TIME=0)`, wird das Paket nur versendet, aber keine Antwort abgewartet. Der maximal mögliche Timeout-Wert ist auf `3600s` festgesetzt.

<TCP><IP=192.168.50.125><PORT=1037><MODE=ASCII><TIME=0><DATA= <SYS><MUTE></SYS> </TCP>

Sendet den System-Befehl `Mute` an das “entfernte” nomos system mit der Adresse `192.168.50.125`

3.6.11 WINDOW

Bei der Verwendung der `WINDOWS` Klasse (nur OS X) ist folgendes zu beachten: WINDOWS werden aus dem SAFARI Browser im KioskMode (Frameless) generiert. Folglich können in dieser Klasse alle Inhalte generiert werden, die sich mit dem SAFARI Browser und deren mögliche PlugIn's anzeigen und steuern lassen.

Ebenso zu beachten ist das „Schichtenmodel" (LAYER). Hierbei werden die WINDOWS nach Prioritäten behandelt. Somit kann gezielt gesteuert werden, welches WINDOW an „oberster" Stelle zur Anzeige gebracht wird. Das Schichtenmodel unterscheidet hierbei zwei Prioritätsgruppen. Dies definiert sich aus dem `LEVEL` und aus dem `SELECT` Befehl.

Wenn `LEVEL` nicht angegeben wird, wird automatisch `LEVEL=NORMAL` angenommen. Durch Level wird der "Raum", in dem sich das Fenster befindet, ausgewählt. Die Auswahl der Fensterreihenfolge innerhalb des "Raums" kann nach wie vor mit Hilfe des `SELECT` -Befehls erfolgen. Ein Fenster mit `SELECT=2` liegt also über einem Fenster mit `SELECT=1`, wenn beide das selbe Level besitzen. Entsprechend liegt ein Fenster mit `SELECT=3` und `LEVEL=HIGH` über(!) einem Fenster mit `SELECT=4` und `LEVEL=NORMAL`, obwohl die ID des Fensters niedriger ist.

Der `LEVEL` -Befehl kann jederzeit (auch bei bereits dargestellten Fenstern) angewendet werden.

Kommando	Argument	Beschreibung
XMIN#	{0...}	X-Koordinate der unteren linken Ecke des Windows (zwingend), `default=0`
YMIN#	{0...}	Y-Koordinate der unteren linken Ecke des Windows (zwingend)
XMAX#	{0...}	X-Koordinate der oberen rechten Ecke des Windows (zwingend), `default=max`
YMAX#	{0...}	Y-Koordinate der oberen rechten Ecke des Windows (zwingend)
LEVEL= `HIGH`: über dem Bildschirmschoner `LOW`: Window unter allen Fenstern, auf dem Desktop-Hintergrund	{LOW,NORMAL,HIGH}	`NORMAL` (default): Window über allen normalen Fenstern, unterhalb des Bildschirmschoners.
Erg. zur Verw. X/YMIN/MAX		Wenn XMIN und YMIN in einer Fensterkonfiguration weggelassen werden, wird das Fenster automatisch zentriert dargestellt. XMIN und XMAX sind mit Default-werten zur Bannerdarstellung in Bildschirmbreite vorbesetzt
SHADOW#	„0" oder „1"	Schatten zeichnen?
TRANS#	{0.0...1.0}	Transparenz des Fensters (zwingend) - Dieser Befehl kann auch nachträglich auf ein bereits dargestelltes Fenster angewendet werden, z.B. zum langsamen Einblenden. default= `0.9`
TITLE#	{Text}	Titel des Fensters
URL#	{Text}	öffnet die URL im Fenster - keine weiteren Kommandos notwendig
IMAGE#	{Text(URL/Pfad)}	zeigt ein lokales oder entferntes Bild an. Das Bild kann entweder als Pfad (z.B. `"/Users/MyH/..."`) oder als URL (`"http://....."`) angegeben werden. Die Regeln für die Darstellung des Bildes gelten analog zum URL-Befehl

Kommando	Argument	Beschreibung
TEXT#	{Text}	stellt TEXT zentriert im Fenster dar – weitere Kommandos notwendig. Eine Pipe im darzustellenden Text erzeugt einen Zeilenvorschub an der entsprechenden Stelle. Die momentan voreingestellte Schriftart für alle Textdarstellungen ist Helvetica-Bold Oblique
TEXTSIZE#	{0.0...1.0}	Textgröße in Pixel (zwingend)
TEXT_RED#	{0.0...1.0}	Rot-Anteil der Textfarbe - default: `1.0`
TEXT_GREEN#	{0.0...1.0}	Grün-Anteil der Textfarbe - default: `1.0`
TEXT_BLUE#	{0.0...1.0}	Blau-Anteil der Textfarbe - default: `1.0`
TEXT_TRANS#	{0.0...1.0}	Transparent-Anteil der Textfarbe - default: `1.0`
BACKGROUND_RED#	{0.0...1.0}	Rot-Anteil des Texthintergrundes - default: `0`
BACKGROUND_GREEN#	{0.0...1.0}	Grün-Anteil des Texthintergrundes - default: `0`
BACKGROUND_BLUE#	{0.0...1.0}	Blau-Anteil des Texthintergrundes - default: `0`
BACKGROUND_TRANS#	{0.0...1.0}	Transparent-Anteil des Texthintergrundes - default: `1.0`
FADETO#	{0..1.0}	blendet die aktuelle Transparenz des Fensters langsam zum angegebenen Wert
CREATE		Abschluss der Konfiguration: erzeugt das Fenster auf dem Bildschirm, wenn nicht geöffnet. (zwingend)
RELEASE		Entfernt das Fenster vom Bildschirm
MINIMIZE		Fenster im Dock minimieren
MAXIMIZE		Fenster im Dock wiederherstellen
SELECT#	{1...10}	Fensterauswahl (1-10) Alle nachfolgenden Befehle beziehen sich immer auf das zuvor mit `SELECT` gewählte Fenster. `SELECT` ist optional (Defaultwert: `1`). Je höher der Wert, desto höher die Priorität. Ein WINDOW mit `SELECT=2` liegt also über dem WINDOW mit `SELECT=1`
RELEASEALL		schließt alle Fenster und setzt `SELECT` auf `1`
RESET		löscht alle Fenster und setzt sämtliche Parameter auf `0`
EFFECTCREATE		wie `CREATE`, aber mit Animation
EFFECTRELEASE		wie `RELEASE`, aber mit Animation
MOVE		bewegt ein bereits dargestelltes Fenster zu den neuen Koordinaten, die zwischenzeitlich gesetzt wurden
EFFECTMOVE		wie `MOVE`, aber mit zusätzlicher Animation

3.6.12 BAOS

Befehlssatz für die Verwendung der BAOS Klasse. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/baos.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel KNX-IP Baos 770

Kommando	Argument	Beschreibung
GETINFO		liefert allgemeine Informationen über den BAOS
GETDESCRIPTION=	{Datenpunktnummer}	liefert die Einstellungen des gewählten Datenpunktes
GETDESCRIPTIONSTRING=	{Datenpunktnummer}	liefert die Beschreibung des gewählten Datenpunktes
GETVALUE=	{Datenpunktnummer}	liefert den aktuellen Wert des Datenpunktes inkl. Zustandsflags im HEX-Format
SETVALUE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	setzt den Wert des gewählten Datenpunkt in HEX, Dezimal oder als ASCII
SENDVALUE=	{DatenpunktNr.}, { HEX/DEC/ASCII}	sendet den Wert auf den KNX-Bus
SETANDSENDVALUE=	{DatenpunktNr.}, { HEX/DEC/ASCII}	Kombination von `SETVALUE` und `SENDVALUE`
READVALUE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	liest den Wert vom KNX-Bus
CLEARTRANSMISSIONSTATE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	löscht den Verbindungsstatus
GETPARAMETERBYTE=	{ParameterNr.}	liefert das Byte des gewählten Parameters

3.6.13 HS

Befehlssatz für die Verwendung des GIRA Homeserver KO-Gateway. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/hs.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel GIRA Homeserver KO-Gateway Support

Kommando	Argument	Beschreibung
SETVALUE#	{Adresse},{Wert}	setzt {Wert} an {Adresse} absolut.
SETVALUEREL#	{Adresse},{Wert}	setzt {Wert} an {Adresse} relativ.
STEPUP#	{Adresse}	erhöht den Wert an {Adresse}.
STEPDOWN#	{Adresse}	senkt den Wert an {Adresse}.
LISTUP#	{Adresse}	erhöht den Wert an {Adresse}.
LISTDOWN#	{Adresse}	senkt den Wert an {Adresse}.

3.6.14 KNX

Befehlssatz für die Verwendung des KNXnet/IP Protokolls (Tunneling/Routing). Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/knx.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel KNXnet/IP Protokoll Support

Kommando	Argument	Beschreibung
SETVALUE=	{Adresse},{Wert}	setzt `{Wert}` an `{Adresse}`
GETVALUE=	{Adresse}	Liest einen Wert einer `{Adresse}`
SCAN=	{Tabellenname}	Scannt eine Liste von Adressen wie in der `knx.csv` definiert
BACKGROUNDSCAN=	{Tabellenname},{ALL}	Wie vor jedoch erfolgt der Scan im Hintergrund bei `{ALL}` werden alle eingetragenen Tabellen gescannt

Zur Verwendung der KNX Klasse müssen die Datenpunkte, die angesprochen werden sollen, in der Definitionsdatei `../misc/knx.esf` vorhanden sein.

3.6.15 STREAMER

Befehlssatz zur Steuerung der Streamer Funktion für die eyeTV Applikation (nur OS X). Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel eyeTV - Streaming Support

Kommando	Argument	Beschreibung
ON		startet die Streaming-Engine und ggf. EyeTV (minimiert)
OFF		beendet die Streaming-Engine, EyeTV läuft weiter
CH#	{Kanalnummer} oder {Kanalname}	wählt die Streaming-Quelle, dieser Befehle funktioniert analog zum `SELECT` -Befehl in der Windows-Klasse. Alle nachfolgenden, kanalbezogenen Befehle beziehen sich auf den zuletzt mit `CH` ausgewählten Kanal
GETCHLIST		liefert eine Liste mit allen zur Zeit verfügbaren Kanälen und der entsprechenden Kanalnummer.
ADDTARGET#	{IP:Port}	weist der zuvor mit `CH` gewählten Quelle ein Ziel zu und beginnt sofort mit dem UDP Streaming zur angegebenen Adresse. Es wird eine eindeutige Streaming-ID zurückgegeben. Auf dem Client dem VLC bitte als URL `udp://@:{Portnummer}` übergeben.
REMOVETARGET#	{IP:Port} oder {Streaming-ID} oder {IP}	stoppt das Streaming des per `IP:Port` oder `ID` angegeben Streams. Wird nur die `IP` angegeben, werden alle an diese `IP` gerichteten Streams gestoppt.
REMOVEALLTARGETS		stoppt alle Streams
GETTARGETSFORIP#	{IP}	listet alle an die `IP` gerichteten Streams auf
GETTARGETSFORCH#	{Kanalnummer} oder {Kanalname}	listet alle `IP-Adressen` auf, an die der Kanal gestreamt wird
GETALLTARGETS		listet alles auf, was gestreamt wird

3.6.16 TIMER

Befehlssatz zur Manipulation der Timer/Kalender Funktionen *(Beispiele)*

Kommando	Argument	Beschreibung
CREATE#	{Name},{Timertyp}, {Wert oder Datums- String}, {Timeraction}	erzeugt einen neuen Timer und schreibt ihn in die `timer.csv`
DELETE#	{Name}	loescht einen existierenden Timer (und stoppt ihn ggfs.)
DISABLE#	{Name}	entfernt `AUTOSTART` aus der `timer.csv` und stoppt den Timer, falls er laufen sollte
ENABLE#	{Name}	schreibt `AUTOSTART` hinter einen existierenden Timer in der `timer.csv` und startet ihn, falls noch nicht geschehen
LIST		listet alle aktiven Timer
LISTALL		listet alle definierten Timer mit ihren Namen auf
RESET#	{Name}	setzt den Timer `{Name}` zurueck
START#	{Name}{Name}	startet den Timer `{Name}`
STOP#	{Name}{Name}	stoppt den Timer `{Name}`
LIST#	{Name}	listet alle aktiven Timer `{Name}`

Timerdefinitionen müssen in der Datei `../misc/timer.csv` definiert sein.

3.6.17 QUICKTIME

Befehlssatz zur Steuerung der Quicktime Applikation (nur OS X)

[QUICKTIME]Kommando	[QUICKTIME]Argument	[QUICKTIME]Beschreibung
CLOSE#	{Fenstertitel}	schliesst das Fenster `{Fenstertitel}`
CLOSEALL		schliesst alle Fenster
FRONT		holt die Quicktime-Applikation nach vorne
FSOFF#	{Fenstertitel}	aktiviert die Tonausgabe
FSON#	{Fenstertitel}	schaltet die Vollbilddarstellung ein
GETALLWINDOWS		liefert eine Liste aller vorhandenen Fenstertitel
GETCTIME#	{Fenstertitel}	liefert die Gesamtspielzeit in Sekunden
GETFRONTWINDOW		liefert den Titel des ersten Fensters
GETPTIME#	{Fenstertitel}	liefert die aktuelle Spielzeit in Sekunden
GETSPEED#	{Fenstertitel}	gibt die aktuelle Wiedergabegeschwindigkeit und -Richtung aus
GETVOL#	{Fenstertitel}	gibt die Wiedergabelautstaerke aus
HIDE		versteckt die Quicktime-Applikation
MUTEOFF#	{Fenstertitel}	aktiviert die Tonausgabe
MUTEON#	{Fenstertitel}	schaltet die Tonausgabe ab
OFF		beendet Quicktime
ON		startet Quicktime
PAUSE#	{Fenstertitel}	unterbricht die Wiedergabe
PLAY#	{Fenstertitel}	spielt die Datei im Fenster
SETFRONTWINDOW#	{Fenstertitel}	holt Fenster `{Fenstertitel}` nach vorne (bei mehreren Fenstern)
SETPTIME#	{Fenstertitel},x	setzt die Spielzeit auf {x} Sekunden
SETSPEED#	{Fenstertitel}, {{-}0.0 ... }	kontrolliert die Wiedergabegeschwindigkeit und Richtung. Bsp: `SETSPEED=Matrix.m4v,-2.5` zweieinhalbfache Geschwindigkeit rückwärts oder: `SETSPEED=Matrix.m4v,3.0` dreifache
SETVOL#	{Fenstertitel}, {0...100}	setzt die Wiedergabelautstaerke
SHOW		zeigt die Quicktime-Applikation
STOP#	{Fenstertitel}	stoppt die Wiedergabe
URL#	{Pfad}}	oeffnet eine Datei im Pfad `{Pfad}`
WINSIZE#	{Fenstertitel}, {x1,y1},{x2,y2}	setzt die Fenstergroesse (und -Position)

Da Quicktime mehrere gleichzeitige Fenster unterstützt, ist bei vielen Befehlen die Angabe des Fenstertitels (entspricht i.d.R. immer dem Dateinamen) notwendig.

3.6.18 Sonderbefehle

Kommando	Argument	Beschreibung
DELAY=	{t} in Sekunden (1 für 1Sek, 0.5 für 500ms, ...)	verzögert die Ausführung der nachfolgenden Befehle um x Sekunden. (Klassenlos!) Bitte immer außerhalb einer Klassensequenz verwenden.
RELAY=	{IP}:{Port}	Die mmh-Sequenz wird per UDP an einen entferntes nomos system weitergeleitet und maximal 3 Sekunden auf eine Antwort gewartet. Beispiel: `<RELAY=192.168.1.50:1037><SYS><ENABLESS></SYS></RELAY>` Die Antwortsequenz hat die Syntax: `<RELAY={IP}:{Port}>{mmh-Antwortsequenz}</RELAY>`
BREAK		beendet sämtliche z.Z. laufenden Kommandosequenzen. (Klassenlos!)

Sonderkommandos sollten nicht innerhalb einer geöffneten Klasse verwendet werden. Diese Befehle sollten immer nach Abschluss- und vor dem Öffnen einer Klasse platziert werden.

Beispiel: `<ITUNES><ON></ITUNES>` <DELAY=3> `<ITUNES><PLAYPLAYLIST=Musik></ITUNES>`

3.6.19 CEC

Befehlssatz für die Steuerung von CEC-fähigen Geräten. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Für die Verwendung von CEC ist keine zus. Konfigurationsdatei notwendig. Bitte beachten, dass die Befehle nicht von allen Geräten unterstützt werden.

Kommando	Argument	Beschreibung
GETDEVICELIST		liefert eine Liste aller am CEC Bus angeschlossenen Geraete, aus der Liste kann man sich ein `{Geraet}` fuer die weiteren Befehle auswaehlen
POWER_ON#	{Geraet}	schaltet `{Geraet}` ein
POWER_OFF#	{Geraet}	schaltet `{Geraet}` aus
TV_SOURCE_TUNER		schaltet den TV auf den lokalen Tuner um
TV_SOURCE_HDMI1		schaltet den TV auf HDMI 1
TV_SOURCE_HDMI2		schaltet den TV auf HDMI 2
TV_SOURCE_HDMI3		schaltet den TV auf HDMI 3
TV_SOURCE_HDMI4		schaltet den TV auf HDMI 4
TV_SOURCE_HDMI5		schaltet den TV auf HDMI 5
TV_SOURCE_HDMI6		schaltet den TV auf HDMI 6
TV_SHOW_MESSAGE#	{Text}	zeigt `{Text}` auf dem TV an (nur von wenigen Geraeten unterstuetzt)
TV_SHOW_POPUP#	{Text}	zeigt `{Text}` auf dem TV an und erwartet Quittierung (nur von wenigen Geraeten unterstuetzt)
AMP_VOLUME_UP		erhoeht die Lautstaerke des AMPs
AMP_VOLUME_DOWN		senkt die Lautstaerke des AMPs
AMP_MUTE_ON		schaltet den Ton des AMPs aus
AMP_MUTE_OFF		schaltet den Ton des AMPs ein
SETPHYSICALADDRESS#	{Adresse}	manuelles Setzen der Adresse des CEC-Adapters
Button Kommandos		
BUTTON_SELECT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_LEFT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RIGHT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RIGHT_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RIGHT_DOWN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_LEFT_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_LEFT_DOWN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_ROOT_MENU#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SETUP_MENU#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_CONTENTS _MENU#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_FAVORITE _MENU#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_EXIT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER0#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER1#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER2#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER3#	{Geraet}	Funktion gem Geraete Dokumentation

Kommando	Argument	Beschreibung
BUTTON_NUMBER4#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER5#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER6#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER7#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER8#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NUMBER9#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_DOT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_ENTER#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_CLEAR#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_NEXT _FAVORITE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_CHANNEL_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_CHANNEL _DOWN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PREVIOUS _CHANNEL#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SOUND_SELECT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SOUND_SELECT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_DISPLAY _INFORMATION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_HELP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAGE_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAGE_DOWN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_POWER#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_VOLUME_UP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_VOLUME _DOWN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_MUTE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PLAY#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_STOP#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAUSE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RECORD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_REWIND#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_FAST _FORWARD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_EJECT#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_FORWARD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_BACKWARD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_STOP_RECORD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAUSE_RECORD#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_ANGLE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SUB_PICTURE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_VIDEO_ON _DEMAND#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_ELECTRONIC _PROGRAM_GUIDE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_TIMER _PROGRAMMING#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_INITIAL _CONFIGURATION#	{Geraet}	Funktion gem Geraete Dokumentation

Kommando	Argument	Beschreibung
BUTTON_PLAY _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAUSE _PLAY_FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RECORD _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_PAUSE _RECORD_FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_STOP _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_MUTE _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_RESTORE _VOLUME_FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_TUNE _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SELECT_MEDIA _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SELECT_AV _INPUT_FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_SELECT_AUDIO _INPUT_FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_POWER_TOGGLE _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_POWER_OFF _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_POWER_ON _FUNCTION#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_F1_BLUE#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_F2_RED#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_F3_GREEN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_F4_YELLOW#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_F5#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_DATA#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_AN_RETURN#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_AN_CHANNELS _LIST#	{Geraet}	Funktion gem Geraete Dokumentation
BUTTON_MAX#	{Geraet}	Funktion gem Geraete Dokumentation

3.6.20 Z-Wave

Befehlssatz zur Steuerung angemeldeter Z-Wave Komponenten

Kommando	Argument	Beschreibung
Allgemein		
RESET		versetzt den zwave-Controller in den Auslieferungszustand und loescht alle angelernten Geraete
INCLUDE#	{ON/OFF}	startet/beendet den Include-Modus (Hinzufuegen von Geraeten)
EXCLUDE#	{ON/OFF}	startet/beendet den Exclude-Modus (Entfernen von Geraeten)
REMOVE#	{Node}	entfernt ein Geraet mit "Gewalt" (z.B. wenn das Geraet defekt ist und nicht mehr excluded werden kann)
DUMPDEVICES		gibt alle angelernten Geraete im Log aus
INTERVIEW#	{Node}	interviewt ein Geraet erneut (findet beim Include implizit statt)
LOADCONFIG#	{Node},{zwaveXML}	laedt eine andere zwave-XML aus der Library fuer das Geraet `{Node}`
SAVE		speichert den aktuellen Zustand des zwave Netzwerks auf der CF-Karte
UPDATENETWORK		aktualisiert die Z-Wave Netzwerk-Topologie, falls Z-Wave Nodes relokalisiert wurden
RESETALARM	{Node}	setzt alle Alarmmeldungen für das Gerät `{Node}` manuell zurück
Device Konfiguration		
CONFMODE#	{Port}	startet den zwave Stack im Konfigurationsmodus (keine `zwave.csv` erforderlich), gueltige Werte fuer `{Port}: /dev/ttyS1, /dev/ttyUSB0, AUTO`
CONFIGURE#	{Node}, {ConfOpt}, {ConfWert}	setzt die Konfigurationsoption `{ConfOpt}` des Geraetes `{Node}` auf `{ConfWert}` , `{ConfOpt}` ist ein Wert von 0-255
GETCONFIGURATION#	{Node}	liefert die Konfigurationswerte des Geraetes `{Node}`
Device Assoziation		
GETASSOCIATIONS#	{Node}	liefert alle Assoziationsgruppen des Geraetes `{Node}` und deren Inhalt
ADDASSOCIATION#	{Node},{Group}, {AddNode}	fuegt das Geraet `{AddNode}` in die Assoziationsgruppe `{Group}` auf dem Geraet `{Node}` ein, `{Group}` ist ein Index
REMOVEASSOCIATION#	{Node},{Group}, {RemoveNode}	entfernt das Geraet `{RemoveNode}` aus der Assoziationsgruppe `{Group}` auf dem Geraet `{Node}`
Wakeup Handling		
GETWAKEUPINTERVAL RANGE#	{Node}	liefert den fuer das Geraet `{Node}` gueltigen Wertebereich fuer den Wakeup (in Sekunden)
GETWAKEUPINTERVAL#	{Node}	liefert den aktuell eingestellten Wakeup-Wert fuer `{Node}` (in Sekunden)
SETWAKEUPINTERVAL#	{Node}	setzt den Wakeup-Wert fuer `{Node}` (in Sekunden)
Befehle		

Kommando	Argument	Beschreibung
SETSWITCH#	{Node}, [{Instanz/CH}], {Wert}	Schaltet den Schalter `{Node}` an bzw. aus, `{Wert}` kann sein: `ON/OFF,0/1,YES/NO`. Wird `[{Instanz/CH}]` verwendet kann auf mögliche weitere Instanzen (zB Mehrkanal Schalter) zugegriffen werden. Wird nur `{Node}` verwendet, wird eine Liste aller Instanzen des Gerätes ausgegeben.
GETSWITCH#	{Node}	liefert den Status des Schalters `{Node}`
SETLEVEL#	{Node}, [{Instanz/CH}], {Wert}	Setzt den Dimmer `{Node}` auf `{Wert}` oder schaltet `{Node}` ein bzw. aus, `{Wert}` kann sein: `0-99, ON/OFF, YES/NO`. Sonst wie `SETSWITCH#`
GETLEVEL#	{Node}	liefert den Wert von `{Node}`
GETTHERMOSTATMODES#	{Node}	liefert alle unterstuetzten Modi des Thermostats `{Node}`
SETTHERMOSTATMODE#	{Node},{Mode}	schaltet den Thermostat `{Node}` auf `{Mode}` um
GETTHERMOSTATSET POINTS#	{Node}	liefert die Setpoints aller Modi des Thermostats `{Node}`
SETTHERMOSTATSET POINT#	{Node},{Mode},{Wert}	aendert den Setpoint des Thermostats `{Node}` im Modus `{Mode}` auf `{Wert}`
RESETMETER#	{Node}	setzt die aufgelaufenen Metering-Werte des Geraets `{Node}` auf 0

3.6.21 Philips HUE

Befehlssatz zur Steuerung angemeldeter HUE Komponenten

Kommando	Argument	Beschreibung
GETALLLIGHTS		liefert eine Liste aller verbundenen Leuchten-Namen
GETALLLIGHTIDS		liefert eine Liste aller verbundenen Leuchten-IDs
RENAME#	{Light},{Name}	benennt `{Light}` in `{Name}` um
POWER_ON#	{Light}	schaltet `{Light}` ein
POWER_OFF#	{Light}	schaltet `{Light}` aus
SETBRIGHTNESS#	{Light},{Wert}	aendert die Helligkeit von `{Light}` auf `{Wert}` - `{Wert}` darf 0..255 sein
SETHUE#	{Light},{Wert}	aendert den Farbton von `{Light}` auf `{Wert}` - `{Wert}` darf 0 - 65535 sein
SETSATURATION#	{Light},{Wert}	aendert die Farbsaettigung von `{Light}` auf `{Wert}` - `{Wert}` darf 0 - 255 sein
SETCOLORTEMP#	{Light},{Wert}	aendert die Farbtemperatur von `{Light}` auf `{Wert}` - `{Wert}` haengt ab von den angesteuerten Leuchten und darf z.Zt. von 153 - 500 sein (6500K - 2000K)
SETEFFECT#	{Light}, COLORLOOP/NONE	fuehrt auf `{Light}` eine ColorLoop aus oder beendet sie
SETALERT#	{Light},SELECT/ LSELECT/NONE	fuehrt auf `{Light}` einen (SELECT) oder mehrere (LSELECT - für 30 Sekunden) "Breathe-Cycle" aus oder beendet sie (NONE)
CREATEGROUP	{group}, {light1}, {light2},..., {lightx}	erzeugt eine neue Gruppe mit dem Namen `{group}` und den nachfolgenden Lampen `{light1}`, `{light2}`...
DELETEGROUP	{group}	löscht die Gruppe `{group}`
GETALLGROUPS		liefert alle in der HUE-Bridge konfigurierten Gruppen
GETGROUPLIGHTS	{group}	liefert eine Liste aller der Gruppe `{group}` zugeordneten Lampen

3.6.22 DEVICE (nomos App Kommando Klasse)

Befehlssatz für die DEVICE Steuerung (nomos APP).

Kommando	Argument	Beschreibung
SETLANGUAGE#	{Lang-Setting}	waehlt die Sprache für die Sprachausgabe
SETVOICE#	{Voice}	waehlt die Stimme für die Sprachausgabe
ANSWER#	{Text}	gibt `{Text}` auf dem Geraet aus, von dem die letzte Spracheingabe kam
SAY#	{Text},{Geraet}	gibt `{Text}` auf allen Geraeten bzw. `{Geraet}` aus
NOTIFY#	{Text},{Geraet}	zeigt einen Hinweistext auf allen Geraeten bzw. `{Geraet}`
ALERT#	{Text},{Geraet}	zeigt einen Alarmtext auf allen Geraeten bzw. `{Geraet}`
RELOAD#	{Geraet}	laedt die GUI auf allen Geraeten oder `{Geraet}` neu
RETURN#	{Geraet}	springt auf allen Geraeten oder `{Geraet}` von der GUI zurueck ins System-Auswahl-Menue
SCREENSAVER#	OFF/0/{Wert}	schaltet den Screensaver `AUS` oder nach `{Wert}` Sekunden ein. Die minimal einstellbare Zeit beträgt 2s. Wird `{Wert} 0` angeben, verhält sich die Funktion wie `AUS`
SETTICKERCONTENT	{ID},{content} [,{value}]	erzeugt einen Screensaver-Ticker `{ID}` mit der Beschreibung `{content}` und dem Wert `{value}`. `{value}` ist optional
REMOVETICKERCONTENT	{ID}	entfernt den Ticker `{ID}`
SHOWTICKERCONTENT	{ID}	Zeigt den Ticker `{ID}` wenn der `SCREENSAVER` aktiv ist

3.7 Dynamische Klassen

Dynamische Klasse betreffen spezielle Gerätetypen, die über einen vordefinierten Befehlssatz verfügen. Zur Zeit unterstützt das nomos system in diesem Bereich den `REMOTE` Befehlssatz für das AppleTV sowie für eine rudimentäre Steuerung von iTunes. Ebenso werden `SONOS` Musik Player unterstützt.

Es ist zu beachten, dass bei Verwendung der Remote Klasse es nach einem Software Update (iTunes Version/ AppleTV Firmware) evtl. zu Fehlfunktionen kommen kann. Wir sind in diesem Fall bestrebt kurzfristig nachzubessern. Eine Nachbesserung findet nur statt, sofern dies möglich ist bzw. der entsprechende Hersteller der Geräte diese Möglichkeit zulässt.

Einen Rechtsanspruch auf diese Eigenschaft wird Seitens der nomos system AG nicht gewährt!

3.7.1 Remote (iTunes/AppleTV)

Befehlssatz für die Steuerung Apple Remote kompatibler Geräte/Software. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/remote.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel Apple Remote Support

Kommando	Argument	Beschreibung
GETINFO		liefert Informationen über den aktuellen Player Status
NEXT		springt zum nächsten Titel
PREV		springt zum vorigen Titel
PAUSE		pausiert den aktuellen Titel
PLAY		spielt den aktuellen Titel ab
TOGGLEPLAY		toggelt zwischen PLAY und PAUSE
STOP		stoppt den aktuellen Titel
SETSHUFFLE#	{ON/OFF}	schaltet den Modus Shuffle der aktuellen Playlist ein o. aus
SETREPEAT#	{ONE/ALL/OFF}	setzt den Repeatmodus der aktuellen Playlist
SETPTIME#	{1...x}	springt zur angegebenen Spielzeit
GETCTIME		liefert die gesamte Spielzeit in Sekunden
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
BUTTON_UP		simuliert einen Tastendruck der AppleRemote
BUTTON_DOWN		simuliert einen Tastendruck der AppleRemote
BUTTON_LEFT		simuliert einen Tastendruck der AppleRemote
BUTTON_RIGHT		simuliert einen Tastendruck der AppleRemote
BUTTON_SELECT		simuliert einen Tastendruck der AppleRemote
BUTTON_SELECTLONG		simuliert einen Tastendruck der AppleRemote (Kontext Menu)
BUTTON_MENU		simuliert einen Tastendruck der AppleRemote
- extended commands -		Erweiterter Befehlssatz, nicht für Verbindungen über Home Sharing
ADDIDTOQUEUE#	{ID}	hängt die ID x an die Up-Next-Queue an
ADDTITLETOQUEUE#	{Name}	hängt den Titel x an die Up-Next-Queue an
ADDARTISTTOQUEUE#	{Name}	hängt den Interpreten x an die Up-Next-Queue an
ADDALBUMTOQUEUE#	{Name}	hängt das Album x an die Up-Next-Queue an
CLEARQUEUE		löscht die Up-Next Queue
DISABLEALLSPEAKERS		schaltet alle Audio-Ausgabegeräte aus
DISABLESPEAKER#	{Speakername}	schaltet `{Speakername}` aus
ENABLEALLSPEAKERS		schaltet alle Audio-Ausgabegeräte ein
ENABLESPEAKER#	{Speakername}	schaltet `{Speakername}` ein
GETALLALBUMS		liefert eine Liste aller Alben
GETALLARTISTS		liefert eine Liste aller Interpreten
GETALLIDSOFPLAYLIST#	{Playlist}	liefert alle IDs in `{Playliste}` (max. 250)
GETALLTITLESOFPLAY LIST#	{Playlist}	liefert alle Titel in `{Playliste}` (max. 250)
GETALLARTISTSOFPLAY LIST#	{Playlist}	liefert alle Interpreten in `{Playliste}` (max. 250)
GETALLALBUMSOFPLAY LIST#	{Playlist}	liefert alle Alben in `{Playliste}` (max. 250)
GETALLPLAYLISTS		liefert eine Liste aller vorhanden Playlisten

Kommando	Argument	Beschreibung
GETIDALBUM#	{ID}	liefert das Album zur angegebenen `{ID}`
GETIDARTIST#	{ID}	liefert den Interpreten zur angegebenen `{ID}`
GETIDTITLE#	{ID}	liefert den Titel zur angegebenen `{ID}`
GETSPEAKERLIST		liefert eine Liste aller erkannten Audio-Ausgabegeräte
GETSPEAKERSTATE#	{Speakername}	liefert den Status (ON bzw. OFF) von `{Speakername}`
GETSPEAKERVOL#	{Name}	liefert die Lautstärke eines Airtunes Lautsprechers relativ zur Master Lautstärke
GETVOL		liefert die aktuelle Applikationslautstaerke
PLAYID#	{ID}	spielt den Titel `{ID}` ab
PLAYPLAYLIST#	{Name}	spielt die Playliste `{Name}` ab
SEARCHALBUM#	{Album}	liefert alle IDs, die zu `{Album}` gehoeren
SEARCHARTIST#	{Suchstring}	liefert eine Liste der Interpreten zurück, auf die der Suchstring passt (Limit: 50)
SEARCHTITLE#	{Suchstring}	liefert eine Track-ID -Liste der Titel zurück, auf die der Suchstring passt (Limit: 50)
SETSPEAKERVOL#	{Name}, {Wert 0-100}	setzt die Lautstärke eines Airtunes Lautsprechers relativ zur Master Lautstärke
SETVOL#	{0...100}	setzt die Lautstärke auf 0-100%

Dazu gibt es die passenden SYS-Befehle:

Kommando	Argument	Beschreibung
GETREMOTEDEVICELIST		liefert eine Liste aller in der `remote.csv` definierten Geräte
REMOTEPAIRING#	{ON}/{OFF}	Startet {ON} /Beendet {OFF} den Pairing Mode für iTunes und AppleTV in der remote Klasse

3.7.2 Sonos

Befehlssatz für die Steuerung von `SONOS` Streaming Playern. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/sonos.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel SONOS Streaming Player Support

Kommando	Argument	Beschreibung
PLAY		spielt den aktuellen Titel ab
PAUSE		pausiert den aktuellen Titel
STOP		stoppt den aktuellen Titel
NEXT		springt zum nächsten Titel
PREV		springt zum vorherigen Titel
JUMPTOINDEX#	{1...x}	springt zum x-ten Titel in der Queue
GETVOL		liefert die aktuelle Applikationslautstärke
SETVOL#	{0...100}	Setzt die Lautstärke auf 0-100%
SETBALANCE#	{-100...0...100}	Setzt die Balance (-100)Links - (+100)Rechts
GETBALANCE		Liefert die aktuelle Balance
MUTEON		Stummschaltung EIN
MUTEOFF		Stummschaltung AUS
LOUDNESSON		Loudness EIN
LOUDNESSOFF		Loudness AUS
SETTREBLE#	{-10...0...10}	EQ Setzt die Höhen
SETBASS#	{-10...0...10}	EQ Setzt die Bässe
SETPTIME#	{Sekunden}	springt zur angegebenen Spielzeit
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
GETCTIME		liefert die gesamte Spielzeit in Sekunden
SETREPEAT#	{ON/OFF,1/0,YES/NO}	schaltet den Wiederholen Modus der aktuellen Playlist ein o. aus
SETSHUFFLE#	{ON/OFF,1/0,YES/NO}	schaltet den Zufall Modus der aktuellen Playlist ein o. aus
SETCROSSFADE#	{ON/OFF,1/0,YES/NO}	schaltet den Überblenden Modus der aktuellen Playlist ein o. aus
GETALLPLAYLISTS		liefert eine Liste aller vorhanden Playlisten
PLAYPLAYLIST#	{Name}	Löscht die Queue und spielt Playliste `{Name}` sofort ab
ENQUEUEPLAYLIST#	{Name}	fügt Album `{Name}` ans Ende der Queue an
GETALLTITLESOFPLAY LIST#	{Name}	gibt alle Titel der Playliste `{Name}` aus
GETALLARTISTSOFPLAY LIST#	{Name}	gibt alle Interpreten der Playliste `{Name}` aus
GETALLALBUMSOFPLAY LIST#	{Name}	gibt alleAlben der Playliste `{Name}` aus
GETALLALBUMS		liefert eine Liste aller Alben
PLAYALBUM#	{Name}	Löscht die Queue und spielt Album `{Name}` sofort ab
ENQUEUEALBUM#	{Name}	fügt Album `{Name}` ans Ende der Queue an
GETALLTITLESOF ALBUM#	{Name}	gibt alle Titel des Albums `{Name}` aus
GETALLARTISTSOF ALBUM#	{Name}	gibt alle Interpreten des Albums `{Name}` aus
GETALLFAVORITES		gibt alle Favoriten aus
PLAYFAVORITE#	{Name}	spielt Favorit `{Name}` sofort ab

Kommando	Argument	Beschreibung
GETALLTITLESOFQUEUE		gibt alle Titel der Queue aus
GETALLARTISTSOFQUEUE		gibt alle Interpreten der Queue aus
GETALLALBUMSOFQUEUE		gibt alle Alben der Queue aus
CLEARQUEUE		löscht die Queue
LEDON		schaltet die LED des Sonos Gerätes ein
LEDOFF		schaltet die LED des Sonos Gerätes aus
ADDMEMBER#	{Name}	fuegt ein weiteres Sonos Gerät mit dem Klassennamen `{Name}` als Wiedergabegeraet hinzu (Quellen-Synchronisierung),"x" muss in der `sonos.csv` definiert sein.
REMOVEMEMBER#	{Name}	entfernt das Sonos Geraet mit dem Klassennamen
STARTLINEINSTREAM#	{Name}	streamt das Signal vom Line-In Eingang an das Gerät mit dem Klassennamen "x", "x" muss in der `sonos.csv` definiert sein
STOPLINEINSTREAM#	{Name}	stoppt den Stream an das Gerät "x"
PLAYLINEIN		spielt das Signal vom Line-In Eingang lokal ab
SETLINEINLEVEL#	{-15 - 15}	-15 - (+)15 - passt die Empfindlichkeit des Line-In Eingangs an die analoge Quelle an
GETLINEINLEVEL		liefert die eingestellte Empfindlichkeit des Line-In Eingangs

3.7.3 XBMC

Befehlssatz für die Steuerung von XBMC Clients. Diese Klasse ist nur verfügbar, wenn das entsprechende AddOn freigeschaltet ist. Zusätzlich muss für das AddOn die entsprechende Definitionsdatei `../misc/ xbmc.csv` definiert sein. Ausführlichere Informationen zu der Anwendung finden Sie im Kapitel SONOS Streaming Player Support

Kommando	Argument	Beschreibung
SHUTDOWN		shutdown system
SUSPEND		suspend system
HIBERNATE		hibernate modus aktivieren
REBOOT		System Neustart
PLAYFILE#	{Datei} oder {URL}	spielt die angegebene Quelle sofort ab
PLAY		spielt den aktuellen Titel ab
PAUSE		pausiert die Wiedergabe
STOP		stoppt die Wiedergabe
NEXT		springt zum naechsten Titel
PREV		springt zum vorherigen Titel
FASTFORWARD		schneller Vorlauf
REWIND		schneller Ruecklauf
BUTTON_UP		simuliert den Tastendruck „Cursor hoch“
BUTTON_DOWN		simuliert den Tastendruck „Cursor runter“
BUTTON_LEFT		simuliert den Tastendruck „Cursor links“
BUTTON_RIGHT		simuliert den Tastendruck „Cursor rechts“
BUTTON_SELECT		simuliert den Tastendruck „Enter“
BUTTON_BACK		simuliert den Tastendruck „Zurueck“
BUTTON_HOME		simuliert den Tastendruck „Home“
BUTTON_INFO		simuliert den Tastendruck „Info“
BUTTON_MENU		simuliert den Tastendruck „Menu“
SETVOL#	{0-100}	Setzt die Lautstaerke auf 0-100%
GETVOL		Abfrage der aktuellen Lautstaerke
SETREPEAT#	{ALL/ONE/ON/OFF}	schaltet den Wiederholen Modus des aktuellen Titel
SETSHUFFLE#		schaltet den Zufall Modus des aktuellen Titel
MUTEON		Stummschaltung EIN
MUTEOFF		Stummschaltung AUS
GETMUTE		Abfrage Status Stummschaltung
SETPTIME#	{Sekunden}	springt zur angegebenen Spielzeit
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
GETCTIME		liefert die gesamte Spielzeit in Sekunden
SHOWMESSAGE#	{Message}	zeigt `{Message}` auf dem Bildschirm an
— Movies —		
GETALLMOVIEIDS		liefert eine Liste alle in der Library vorhandenen Movie-IDs
PLAYMOVIEID#	{Movie-ID}	spielt `{Movie-ID}` ab
GETMOVIEIDTITLE#	{Movie-ID}	liefert den Titel zur `{Movie-ID}`
— PVR-Channels —		
GETALLCHANNELIDS		liefert eine Liste aller Channel-IDs
PLAYCHANNELID#	{Channel-ID}	schaltet auf den Sender mit `{Channel-ID}` um
GETCHANNELIDNAME#	{Channel-ID}	liefert den Sendernamen zur `{Channel-ID}`

Kommando	Argument	Beschreibung
GETINFO		liefert Status-Info (identisch zum Broadcast)
— Interpreten —		
GETALLARTISTIDS		liefert eine Artist-ID-Liste aller Interpreten
GETARTISTIDNAME#	{Artist-ID}	liefert den Interpreten zur `{Artist-ID}`
PLAYARTISTID#	{Artist-ID}	spielt alle Songs von `{Artist-ID}` ab
GETSONGIDSOFARTISTID#	{Artist-ID}	liefet eine Song-ID-Liste aller Songs zur `{Artist-ID}`
— Alben —		
GETALLALBUMIDS		liefert eineAlbum-ID-Liste aller Alben
GETALBUMIDNAME#	{Album-ID}	liefert den Albumnamen zur `{Album-ID}`
PLAYALBUMID#	{Album-ID}	spielt das Album mit der `{Album-ID}` ab
GETSONGIDSOFALBUMID#	{Album-ID}	liefet eine Song-ID-Liste aller Songs des Albums `{Album-ID}`
— Songs —		
GETALLSONGIDS		liefert eine Song-ID-Liste aller Songs
GETSONGIDTITLE#	{Song-ID}	liefert den Titel zur `{Song-ID}`
GETSONGIDARTIST#	{Song-ID}	liefert den Interpreten zur `{Song-ID}`
GETSONGIDALBUM#	{Song-ID}	liefert das Album zur `{Song-ID}`
PLAYSONGID#	{Song-ID}	spielt `{Song-ID}` sofort ab
— Audio-Playliste —		
GETSONGIDSOFAUDIO PLAYLIST		liefert eine Liste aller `{Song-IDs}` in der Audio-Playliste (enspricht der Audio-Queue)
PLAYAUDIOPLAYLIST		Spielt die Audio-Playliste ab
CLEARAUDIOPLAYLIST		loescht die Audio-Playliste
ADDSONGIDTOAUDIO PLAYLIST#	{Song-ID}	fuegt `{Song-ID}` zur Audio-Playliste hinzu
ADDALBUMIDTOAUDIO PLAYLIST#	{Album-ID}	fuegt alle Titel von `{Album-ID}` zur Audio-Playliste hinzu
ADDARTISTIDTOAUDIO PLAYLIST#	{Artist-ID}	fuegt alle Titel von `{Artist-ID}` zur Audio-Playliste hinzu
— Video-Playliste —		
GETMOVIEIDSOFVIDEO PLAYLIST		liefert eine Liste aller `{Movie-IDs}` in der Video-Playliste (entspricht der Video-Queue)
PLAYVIDEOPLAYLIST		Spielt die Video-Playliste ab
CLEARVIDEOPLAYLIST		loescht die Video-Playliste
ADDMOVIEIDTOVIDEO PLAYLIST		fuegt `{Movie-ID}` zur Video-Playliste hinzu
— Suchfunktionen —		
SEARCHTITLE#	{Titel}	sucht `{Titel}` und liefert eine Song-ID-Liste aller Treffer (Anfangsbuchstaben ausreichend)
SEARCHARTIST#	{Interpret}	sucht `{Interpret}` und liefert eine Artist-ID-Liste aller Treffer (Anfangsbuchstaben ausreichend)
SEARCHALBUM#	{Album}	sucht `{Album}` und liefert eine Album-ID-Liste aller Treffer (Anfangsbuchstaben ausreichend)
SEARCHMOVIE#	{Filmtitel}	sucht `{Filmtitel}` und liefert eine Movie-ID-Liste aller Treffer (Anfangsbuchstaben ausreichend)

3.8 Tastaturbelegung (Sondertasten)

Für die Bearbeitung der folgend beschriebenen Definitionstabellen, werden teilweise Zeichen benötigt, die auf dem Tastaturlayout des Mac Systems nicht vorhanden sind. Nachfolgende Tabelle liefert einen Auszug der notwendigen und wichtigen Zeichen sowie deren Aufruf:

<table>
<tr><td>@</td><td>MacOS = [ALT]+L</td></tr>
<tr><td>|</td><td>MacOS = [ALT]+7</td></tr>
<tr><td>[</td><td>MacOS = [ALT]+5</td></tr>
<tr><td>]</td><td>MacOS = [ALT]+6</td></tr>
<tr><td>{</td><td>MacOS = [ALT]+8</td></tr>
<tr><td>}</td><td>MacOS = [ALT]+9</td></tr>
<tr><td>\</td><td>MacOS = [ALT]+[SHIFT]+7</td></tr>
<tr><td>-</td><td></td></tr>
<tr><td>@</td><td>WinOS = [CTRL]+[ALT]+Q</td></tr>
<tr><td>|</td><td>WinOS = [CTRL]+[ALT]+<</td></tr>
<tr><td>[</td><td>WinOS = [CTRL]+[ALT]+8</td></tr>
<tr><td>]</td><td>WinOS = [CTRL]+[ALT]+9</td></tr>
<tr><td>{</td><td>WinOS = [CTRL]+[ALT]+7</td></tr>
<tr><td>}</td><td>WinOS = [CTRL]+[ALT]+0</td></tr>
<tr><td>\</td><td>WinOS = [CTRL]+[ALT]+ß</td></tr>
</table>

Das Apple Keyboard unterscheidet bei der Tatstatur zwischen der `[ALT]` Taste **Links** und der `[ALT]` Taste **Rechts** neben der Leertaste (Space). So kann bei Verwendung der rechten `[ALT]` Taste unter WINDOW auf das zusätzlich Betätigen der `[CTRL]` Taste verzichtet werden.

KAPITEL 4

Erweiterte Funktionen (AddOn's)

Die CommandServer Erweiterung
GIRA Homserver KO-Gateway, KNXnet IP Support
Event Server
eyeTV - Streaming Support
Apple Remote Support
Sonos Streaming Player Support
Z-Wave Support
Philips HUE Support
mremote Support (commandFusion iViewer support)

Das nomos system ist modular aufgebaut und kann funktional nahezu beliebig erweitert werden. Ebenso können über sogenannte CommandServer Addons auch eigene Protokollschnittstellen definiert und eingebunden werden. Komplexere Schnittstellen wie zB für KNX, Z-Wave, SONOS oder dem Control System Protokoll von Commandfusion sind als fertiges Modul verfügbar. Bitte beachten Sie, dass für die Verwendung der Module in der Regel auch eine entsprechende Lizenzierung notwendig ist.

Alle diese Module erweitern den nomos system Befehlssatz entsprechend und ermöglichen die Verwendung des nomos systems als eigenständige Steuerzentrale oder auch als multi Protokoll Gateway, welches alle angebundenen Systeme über ein einheitliches, rudimentäres ASCII Protokoll miteinander kommunizieren lässt.

All diese Module/Schnittstellen sind wie nachfolgend beschrieben offen gelegt und können durch den Nutzer beliebig verändert, angepasst und auch erweitert werden.

Die nachfolgend beschriebenen Definitionsdateien werden im `.csv` Format erstellt. Es ist auf strikte Einhaltung der jeweiligen Konfigurationsoptionen für die jeweiligen Dateien zu achten.Trennzeichen für die jeweiligen Funktionen innerhalb einer Zeile ist das Semikolon. Das Ende der Definitionszeile sollte ebenfalls mit einem Semikolon abgeschlossen werden. Es sollte beachtet werden, dass das Zeilenende ordentlich abgeschlossen wird (Line Feed oder Return).

Es empfiehlt sich, die Dateien ausreichend zu dokumentieren. Zu diesem Zweck können Kommentartexte am Zeilenanfang oder am Zeilenende verwendet werden.

// Dies ist ein Kommentar Eine einfacher Kommentartext am Anfang der Zeile

1;1;<SYS><RESTART></SYS>; // Neustart des Daemons Ein einfacher Kommentartext am Ende der Zeile. Hierbei ist zu beachten, dass vor dem Kommentar Marker „//“ mindestens ein Tabulator gesetzt wird.

/*

Unterstützte EIS Typen in der esf Datei

EIS 1 'Switching' (1 Bit)

EIS 2 'Dimming - position' (1 Bit)

EIS 2 'Dimming - control' (4 Bit)

EIS 2 'Dimming - value' (8 Bit)

EIS 3 'Time' (3 Byte)

EIS 4 'Date' (3 Byte)

EIS 5 'Value' (2 Byte)

***/**

mehrzeiligen Kommentare werden mit `"/*"` und `"*/"` eingefasst.

Am Ende der Dokumentation sind sämtliche Definitionsdateien als Vorlage angehangen.

Sollten die Module nach deren Konfiguration nicht wie gewünscht arbeiten, empfiehlt sich die Betrachtung des Startlogs. Das Log gibt Auskunft, ob die jeweiligen Dateien korrekt eingelesen werden konnten und das entsprechende Modul aktiviert wurde.

Beispiel für die Startlog Ausgabe der `remote.csv`

```
15:08:35.913 - system: ### remote ###
15:08:35.913 - system: reading path: /root/nomos/projects/Nomos_LB/misc
15:08:35.913 - system: reading file: remote.csv
15:08:35.914 - system: file remote.csv has 3 valid lines
15:08:35.914 - system: (remote.csv) section [DEVICES] has 2 entries
15:08:35.914 - remote: read 2 remote device definitions
```

Beispiel für die Startlog Ausgabe eines CommandServers (`LG1.csv`)

```
15:08:32.867 - system: ### commandservers ###
15:08:32.867 - system: reading path: /root/nomos/projects/Nomos_LB/addons
15:08:32.867 - system: reading file: LG1.csv
15:08:32.874 - system: file LG1.csv has 50 valid lines
```

4.1 Die CommandServer Erweiterung

Aufbau der CommandServer Definitionstabellen

Reservierte (extended) Servertypen

Verwendung von USB RS232 Umsetzern

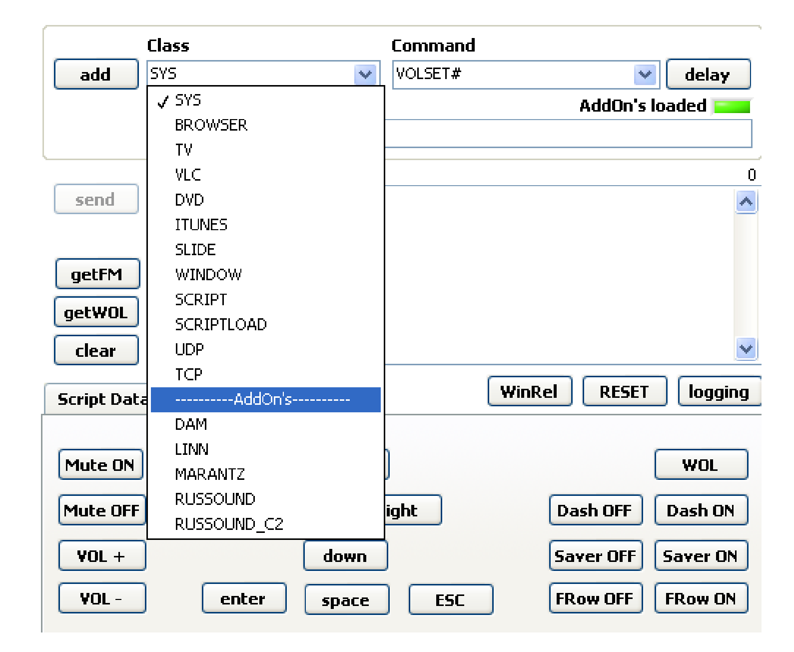
Der CommandServer dient dazu Protokolle zusätzlicher Geräte in den nomos system Befehlssatz zu implementieren. So ist es z.B. notwendig neben dem Steuern des nomos systems auch zusätzliche Peripherien wie z.B. ein Multiroom Audiosystem, ein mögliches TV System, einen Projektor oder einen IR Emitter in die Steuerung einzubinden. Das nomos system kann somit auch als komplexe Steuerzentrale eingesetzt werden. Vor allem wird durch diese Technologie dem Anwender die Integration solcher Systeme erheblich vereinfacht, da er sich lediglich mit dem nomos system Befehlssatz auseinander setzen muss. Bereits verfügbare CommandServer Definitionsdateien werden auf unserer Hompage www.nomos-system.com zum freien Download bereit gestellt. Das Angebot an Definitionsdateien wird permanent erweitert.

Die nomos system Grundlizenz (Gate) verfügt über eine CommandServer Lizenz. Dies bedeutet, dass ein Fremdgerät gesteuert werden kann, also eine CommandServer Definitionsdatei verarbeitet wird. Die nomos System Gate+ Lizenz verfügt über 10 CommandServer Lizenzen und kann bis auf max. 25 CommandServer Lizenzen (nomos Box und Mac OSX) ausgebaut werden.

Das nomos system prüft beim Start des Services, ob Definitionsdateien verfügbar sind. Sofern entsprechende Einträge gefunden wurden, werden die Befehlsketten eingebunden und stehen der Steuerung zur Verfügung.

Bitte achten Sie darauf, dass die Anzahl der Definitionstabellen in dem entsprechenden Verzeichnis die Anzahl der lizenzierten CommandServer nicht überschreitet. nomos implementiert die Tabellen sequentiell. D.h. die Tabellen werden in der Reihenfolge gelesen, wie sie vom Dateisystem aufgrund der Namensgebung zur Verfügung gestellt werden. Ist die Anzahl der Tabellen nun größer als die Anzahl der Lizenzen, werden weitere Tabellen ignoriert. Auskunft über die eingelesenen Dateien gibt das Startlog.

Bitte achten Sie bei der Wahl des .csv-Namens darauf, keine bereits existierenden Klassenbezeichnungen (SYS, BROWSER, ITUNES,TV, ...) zu verwenden, damit eine eindeutige Zuordnung der CommandServer- Klasse gewährleistet ist



Nach dem Einlesen der Definitionstabellen erweitern sich die Befehlsklassen um den Inhalt der Definitionstabellen.

Die Ausgabe aller verfügbaren CommandServer kann auch per Befehl eingeleitet werden:

<SYS><GETSRVLIST></SYS> Abfrage der aktuell verfügbaren CommandServer Klassen

Antwort:

<SYS>GETSRVLIST=IRT_SAM|NUVO|PROWL|TEMP|YAHOO|OK<SYS> Die beispielhafte Antwort besagt, das 5 zus. Klassen verfügbar sind. Die Kommandos der jeweiligen Klassen sind ebenfalls abrufbar.

Sind die CommandServer Klassen bekannt, können ebenfalls die jeweiligen Befehle abgefragt werden. Je CommndServer Klasse steht der Befehl <[CLASS]><GETCMDLIST></[CLASS]> zur Verfügung.

<IRT_SAM><GETCMDLIST></IRT_SAM> Abfrage der aktuell verfügbaren Kommandos der CommandServer Klasse <IRT_SYM>

Antwort: <IRT_SAM>GETCMDLIST=POWER_TOGGLE | OFF | ON | VOL- | VOL+ | TOGGLE_MUTE | PROG- | PROG+ | P.MODE | S.EFFECT | S.MODE | SOURCE_TOGGLE | TV | AV | S-VID | HDMI1 | HDMI2 | EXT1 | EXT2 | COMP | PC | WISELING_TOGGLE | TEXT | KEY_-/-- | KEY_00 | KEY_01 | KEY_02 | KEY_03 | KEY_04 | KEY_05 | KEY_06 | KEY_07 | KEY_08 | KEY_09 | KEY_BLUE | KEY_GREEN | KEY_YELLOW | KEY_RED | KEY_UP | KEY_DOWN | KEY_LEFT | KEY_RIGHT | KEY_MENU | KEY_EXIT | KEY_RETURN | PLAY_PAUSE | STOP | REW | FF | HELP | PIP | CH.MGR | MOVE | P<P | SRS | STILL | TIMER1 | OK</IRT_SAM>

Die beispielhafte Antwort gibt eine Liste aller verfügbaren Kommandos dieser Klasse aus. Enthält ein Kommando am Ende ein „#" .wie zB `...|VOLUME#|REW|...` bedeutet dies, dass dem Kommando ein Wert folgen muss `<[CLASS]><VOLUME=23></[CLASS]>`.

4.1.1 Aufbau der CommandServer Definitionstabellen

Wie bereits beschrieben, bieten CommandServer die Möglichkeit beliebige Systeme an das nomos system anzubinden. Hierbei arbeitet der CommandServer auch gleichzeitig als Protokollwandler, der das spezifische Protokoll des Fremdsystems in die nomos system Protokollstruktur integriert. Der CommandServer arbeitet bidirektional. Er verarbeitet also eingehende wie ausgehende Protokollsequenzen.

Durch den einfachen Aufbau der Definitionsdateien ist ein höchst effizientes Einbinden von Fremdprotokollen gewährleistet. Es können nahezu alle Protokolltypen verarbeitet werden.

Die CommandServer Definitionsdateien werden beim Start des nomos systems eingelesen und bereitgestellt. Sollte kein Startup Kommando (`STARTUPCMD`) definiert sein, erfolgt ein Verbindungsaufbau zu dem Zielsystem erst nach dem Absetzen einer ersten Befehlssequenz. Sollen Meldungen (Events) von den Fremdsystemen verarbeitet werden ist es notwendig, die Verbindung dauerhaft herzustellen. Dies kann zB über den Parameter `TIMEOUT/TIMEOUT_IN_MS` entsprechend manipuliert werden. Die `[CONFIG]` Sektion verfügt über sehr viele weitere Möglichkeiten, auf das Verhalten der Kommunikationsschnittstelle individuell Einfluss zu nehmen.

Mögliche bzw. verfügbare Parameter der Definitionsdatei werden ebenfalls im Startlog ausgegeben, sofern die `DEBUG` Option gesetzt ist. Die Ausgabe dient der Information und Hilfestellung bei fehlerhaftem Verhalten.

```
21:46:57.310 - system: ### commandservers ###
21:46:57.310 - system: reading path: /Users/admin/Documents/Nomos_Test/addons
21:46:57.310 - system: reading file: cserver.csv
21:46:57.315 - system: file cserver.csv has 7 valid lines
21:46:57.315 - system: (cserver.csv) NOTICE: LOCALPORT not found
21:46:57.315 - system: (cserver.csv) NOTICE: SPACING not found
21:46:57.315 - system: (cserver.csv) NOTICE: STARTUPCMD not found
21:46:57.315 - system: (cserver.csv) NOTICE: HEARTBEATCMD not found
21:46:57.315 - system: (cserver.csv) NOTICE: RECONNECT not found
21:46:57.316 - system: (cserver.csv) NOTICE: SERVERTYPE not found
21:46:57.316 - system: (cserver.csv) NOTICE: CMDPREFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: CMDSUFFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: MAPPREFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: MAPSUFFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: SEQUENCEPREFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: SEQUENCESUFFIX not found
21:46:57.316 - system: (cserver.csv) NOTICE: STARTDELIMITER not found
21:46:57.316 - system: (cserver.csv) NOTICE: ENDDELIMITER not found
21:46:57.316 - system: (cserver.csv) NOTICE: CLIENTPORT not found
21:46:57.316 - system: (cserver.csv) NOTICE: USERNAME not found
21:46:57.316 - system: (cserver.csv) NOTICE: PASSWORD not found
21:46:57.316 - system: (cserver.csv) NOTICE: MATCHING not found
21:46:57.316 - system: (cserver.csv) NOTICE: section [COMMANDS] has no entries
21:46:57.316 - system: (cserver.csv) NOTICE: section [MAPPINGS] has no entries
21:46:57.316 - system: (cserver.csv) NOTICE: EXPORT not found
```

Wird ein CommandServer vom nomos system abgewiesen, wird dies ebenso im Startlog ausgegeben.

```
22:25:48.869 - system: (cserver.csv) ERROR: SERVERMODE not found
22:25:48.869 - system: ERROR: config of cserver.csv is not valid - skipping
```

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.{projectPath}addon
Name:	[CLASS].csv
Länge:	2000 Zeilen/Einträge je Sektion
Event:	-
Export:	Ja
Lizenz:	Ja

Die `[CLASS].csv` **besteht aus 3 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[COMMAND]; Definition der jeweiligen Protokollsequenz und Zuordnung des Kommandonamen

[MAPPINGS] Event Definitionen möglicher Antwortsequenzen

Aufbau:

Im `\{projectPath}\addon\` Ordner eine `{ClassName}.csv` mit folgendem Inhalt anlegen. Der Dateiname definiert den Klassennamen. Es ist darauf zu achten, dass der verwendete Klassenname dem nomos system noch nicht bekannt ist:

- **[CONFIG];Sektion**

  ACTIVE; Aktiviert/Deaktiviert den Webserver
  „`YES`" (Default)= aktiv,

  DEBUG; Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

  SERVERIP; `{Device-IP};` oder `LOCAL` bei USB/RS232 Betrieb. Dieser Parameter ist zwingend erforderlich.

  SERVERPORT; `{TCP/UDP/AUTO Device-Port};` oder bei Verwendung von USB/RS232 Adaptern `AUTO` oder Serieller Port; `{Geschwindigkeit}; {Datenbits}{Parität}; {Stopbits}; {Flußkontrolle}.` Dieser Parameter ist zwingend erforderlich. bei seriellen Commandservern mit `SERVERPORT;AUTO` werden zuerst die Hardware-Ports verwendet, sofern vorhanden

  SERVERPROTOCOL; Protokoll: `"UDP"`, `"TCP"` , `"HTTP"` , `"HTTPS"` oder „`SERIAL`" für USB/RS232 Adapter. Default = `"UDP"`

  SERVERMODE; Übertragungsart: `"ASCII"` oder `"HEX"` . DieserParameter ist zwingend erforderlich.

  SERVERTIMEOUT_IN_MS; Zeit (in ms), die auf eine möglich Antwort gewartet wird. `0` = nicht warten, `NONE` = Verbindung dauerhaft halten Wenn `SERVERTIMEOUT;NONE` definiert wurde, wird nach dem Absetzen eines Befehls die Verbindung dauerhaft aufrecht erhalten und eintreffende Daten des Gerätes interpretiert (vgl. Eventserver). Dies funktioniert mit UDP,TCP und seriellen Verbindungen. Der Timeout lässt sich auch weiterhin kommandobezogen definieren.

  MATCHING; Einstellung der Abhandlung für das Parsen der `"MAPPINGS NORMAL"` (Default) oder „`FULL`"

  EXPORT CommandServer Freigabe/Sharing aktiv
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

  CLIENTIP; `{Device-IP}:` gibt eine weitere IP an, an die das Reply des Commandservers gesendet wird.

CLIENTPORT; `{Device-PORT}` **:** gibt den Port der `CLIENTIP` an, an die das Reply gesendet werden soll. `CLIENTIP` und `CLIENTPORT` müssen beide definiert sein

LOCALPORT; Manuelle Vorgabe des lokalen Quellport für die Verbindungen. Mögliche Werte: `RANDOM` oder `1..65535` . Wenn `LOCALPORT` nicht definiert ist, ist der lokale Port automatisch entsprechend dem `SERVERPORT` (wie bisher). Falls der `Quellport` bereits genutzt wird, wird automatisch ein zufälliger Quellport gewählt und ein Logeintrag geschrieben

SERVERTIMEOUT; Wie `SERVERTIMEOUT_IN_MS` , jedoch Zeit (in s), die auf eine möglich Antwort gewartet wird. Wenn `SERVERTIMEOUT` und `SERVERTIMEOUT_IN_MS` gleichzeitig definiert sind, hat die Einstellung für `SERVER-TIMEOUT_IN_MS` Vorrang.

SERVERTYPE; Definition der Checksummen Regel (HEX CommandServer), die bei Verwendung entsprechend berücksichtigt wird. Ist eine Regel definiert, wird die Protokollsequenz entsprechend ergänzt. Zu den frei definierbaren Regeln existieren auch fest eingebaute Regeln für spezifische Protokolle.

RUS für das Russound RNet Protokoll

SPEAKERCRAFT für die gleichnamigen Produkte.

ADNOTAM für die gleichnamigen Produkte.

SUM= `{Index Startbyte}-{Index Endbyte}` berechnet das Lowbyte der Summe von `{Index Startbyte}` bis `{Index Endbyte}` und fügt es ans Ende der Protokollsequenz an. Die Indizes sind optional. Die Option kann nur im HEX-Modus verwendet werden.

Beispiele: (Ausgangssequenz: 0102030405)

-	SUM	: 01020304050F
-	SUM=1	: 01020304050F
-	SUM=2	: 01020304050E
-	SUM=1-5	: 01020304050F
-	SUM=2-4	: 010203040509

STARTUPCMD; führt `{Kommando}` einmalig beim Start des CommandServers aus. Die verwendeten Kommandos müssen unter `[COMMANDS];` definiert sein. Mehrere Befehle können mittels „,“ aneinandergereiht werden. Z.B: `{Befehl1},{Befehl2},{Befehl3}`.
Für die verwendeten Befehle gelten alle Regeln (Prefix, Suffix, Checksumme usw.), die ggfs. vorhanden sind.
Beispiel:
`STARTUPCMD;POWER_ON,STATE`
Sendet beim Start des CommandServers Die Befehle `POWER_ON` und `STATE`, wie unter `[COMMANDS]` definiert.

HEARTBEATCMD; `{HeartbeatCMD};{Heartbeatinterval};`
Führt das Kommando `{HeartbeatCMD}` in der Intervallzeit `{Heartbeatinterval}` im Sekunden Intervall aus, wenn für `{Heartbeatinterval}` keine Zeit angegeben wurde, gilt die default Einstellung von 60s.

Für das `{HeartbeatCMD}` gelten die Bedingungen wie für `STARTUPCMD` . Es kann jedoch nur ein Kommando eingestellt werden. Das `HEARTBEATCMD` wird nur ausgeführt, wenn eine Verbindung zu dem Zielsystem besteht.

RECONNECT stellt die Verbindung nach fehlgeschlagenem Senden einmalig wieder her.
„`YES`" = aktiv,
„`NO`" (Default) = nicht aktiv
Voraussetzungen:
a) STARTUPCMD ist definiert
b) Es ist eine dauerhafte Verbindung hergestellt worden

CMDPREFIX; ergänzt das/die Kommando/Protokollausgabe um entsprechende Zeichen am Anfang der Zeichenkette. Nicht zu verwenden, wenn `SERVERTYPE` gesetzt.

CMDSUFFIX; ergänzt das/die Kommando/Protokollausgabe um entsprechende Zeichen am Ende der Zeichenkette. Nicht zu verwenden, wenn `SERVERTYPE` gesetzt.

SEQUENZPREFIX wie `CMDPREFIX` wirkt jedoch am Anfang der gesamten Protokollsequenz. Nicht zu verwenden, wenn `SERVERTYPE` gesetzt.

SEQUENZSUFFIX wie `CMDSUFFIX` wirkt jedoch am Ende der gesamten Protokollsequenz. Nicht zu verwenden, wenn `SERVERTYPE` gesetzt.

STARTDELIMITER, ENDDELIMITER `{Zeichen - oder Bytesequenz}`
Die Delimiter greifen sofort beim Empfang einer (Teil-)Sequenz und reichen die komplette Empfangssequenz (die aus mehreren Teilsequenzen bestehen kann) erst dann an die Matching-Engine weiter, wenn die Bedingung(en) erfüllt sind. Die Logausgaben sind diesbezüglich erweitert worden. Sollte der Standard-Empfangspuffer (8K) zu klein sein, wird er dynamisch in 8K-Schritten vergrössert. Ebenfalls wird sichergestellt, dass bei einer empfangenen Sequenz, die die Delimiter-Bedingungen mehrfach erfüllt, jeder Teil separiert an die Matching-Engine weitergegeben wird. Es reicht aus, einen der beiden Delimiter anzugeben.

MAPPREFIX; fügt vor jedem Mapping-Eintrag entsprechend Zeichen ein (analog zu `CMDPREFIX`)

MAPSUFFIX; fügt jedem Mapping-Eintrag entsprechend Zeichen an (analog zu `CMDSUFFIX`)

SPACING; Legt den Mindestabstand `{Zeit in ms}` zwischen 2 Kommandosequenzen fest. Begrenzt somit den Sendezyklus zwischen den Protokollsequenzen auf einen definierten Wert. Dies kann zB bei dem Absetzen von IR Befehlen hilfreich sein.

USERNAME; Anmeldung Username, sofern notwendig

PASSWORD; Anmeldung Password, sofern notwendig

ENCRYPTION; reserviert. zZ ohne Funktion

Beispiel für die Verwendung der xxxPREFIX/SUFFIX Optionen:

`CMDPREFIX;`A ;

`CMDSUFFIX;`E

```
SEQUENZPREFIX;X
SEQUENZSUFFIX;Z

[COMMANDS];
POWER_ON;01 ;
HDMI1;02 ;
SC_DIGITAL;03 ;
```

Folgende beispielhafte Kommandosequenz:
`<{CLASS}><POWER_ON><HDMI1><SC_DIGITAL></{CLASS}>`

Erzeugt folgende Ausgabe: X A 01 EA 02 EA 03 E Z

- **[COMMANDS];Sektion - Kommando-Definitionen**

  Aufbau der Kommandodefinition:
  {CMD};{Sequenz};{Command Timeout[ms]};{Paramteter};

  Die Kommando Definitionen weisen einem frei definierbaren Kommando `{CMD}` eine zu versendende Sequenz `{Sequenz}` zu, die im entsprechenden Format `(ASCII/HEX)` angegeben wird. In `{Sequenz}` können folgende Spezialzeichen vorkommen:

  "\#" fügt das mit {CMD} übergebene Argument an der entsprechenden Stelle in `{Sequenz}` ein.

  Im ASCII-Modus wird das Argument ohne weitere Wandlung eingefügt; Im HEX-Modus wird das Argument vorzeichenbehaftet (1 Byte) umgewandelt und anschließend eingefügt. Mehrere Argumente können durch Kommata getrennt übergeben werden.

  Die Sequenz wird in der Reihenfolge der übergebenen Kommandos aufgebaut und nach Terminierung der Klasse abgesendet.

  {CMD}; Frei definierbarer Kommando Name unter dem das Senden der Protokollsequenz ausgelöst wird.

  {Sequenz}; Zugeordnete Protokollsequenz die an das Zielsystem übertragen wird, wenn das Kommando entsprechend aufgerufen wird.

  {Command Timeout[ms]} Timeout Option, die jedem Kommando individuell zugeordnet werden kann. Das Setzen dieser Option ignoriert die globale Timeout Einstellung in der `[CONFIG]` Sektion. Bei einer Anreihung von Kommandos innerhalb einer Sequenz, gilt das Kommando Timeout, sofern definiert, des letzten Befehls.

  {Paramteter}; Einstellung zusätzlicher Option für die Manipulation der Wertübergabe `"\#"` . Siehe auch Formatoptionen oder Bedingungen/Vergleicher.

Beispiele für mögliche Definitionen [COMMANDS]

SERVERMODE; ASCII SWITCH;SCHALTE_AUF_DEN_\#-KANAL RECORD;_AND_RECORD_\#_AND_\#

Die Befehle: <[CLASS]><SWITCH=ZDF><RECORD=ARD,RTL><[/CLASS]>

erzeugen folgende Sequenz: „SCHALTE_AUF_DEN_ZDF-KANAL_AND_RECORD_ARD_AND_RTL"

Dem ASCII String können auch nicht darstellbare Zeichen übergeben werden. Diese Zeichen werden ebenso eingebunden wie ein Argument. Bitte beachten Sie, dass jeweils nur ein Zeichen übergeben werden kann.

Diese Zeichen werden wie folgt übergeben: `SWITCH;SCHALTE_AUF_DEN_\#-KANAL\0x0D\0x` ist hierbei der Platzhalter für folgende HEX Bytes, `0D` ist das zu sendende Byte (in diesem Fall ein Zeilenvorschub bzw. ein RETURN). Bitte beachten Sie, dass je Byte der Platzhalter zu verwenden ist. Also `\0x0D\0x0C\0xFF` ... usw.

SERVERMODE; HEX SWITCH;07AB\#04; RECORD;07CF\#BD\#06;

Die Befehle: <[CLASS]><SWITCH=1><RECORD=-1,2></[CLASS]>

erzeugen folgende Sequenz: 07AB010407CFFFBD0206

SERVERMODE; HTTP/HTTPS PLAY;/cgi-bin/vcontrol?command=play; STOP;/cgi-bin/vcontrol?command=stop;

Die Befehle: <[CLASS]><PLAY></[CLASS]>

erzeugen folgende Sequenz: GET /cgi-bin/vcontrol?command=play HTTP/1.1

Der Servermode HTTP dient z.B. zur Ansteuerung von Geräten, die über eine CGI Schnittstelle bzw. ein WEB Interface verfügen. Ebenso können hierüber Webabfragen generiert werden.

Sobald Formatoptionen für einen Befehl definiert sind und der HEX-Modus aktiv ist, wird die implizite automatische Argument-Umwandlung für diesen Befehl abgeschaltet. Sollte dennoch eine Hex-Wandlung gewünscht werden, ist der Formatierer %hex zu verwenden.

Verschiedene Beispiele für die Verwendung von Formatoptionen

VOLUME;<SYS><SETVOL=\#></SYS>;;{%-20} Zieht von jedem übergebenen (Dezimal-)Argument 20 ab, bevor das Argument in die Sequenz eingesetzt wird.

TEMPSET;!SET_C\#;20;{%*1.00} Skaliert den Übergabewert auf zwei Nachkomma Stellen. zusätzlich ist noch ein Kommando Timeout von 20ms definiert.

CFG_INITVOL(V);INIVOL\#\x0D;;{%-100,%*-1,%/1.2658,%*1} Skaliert eine Wertübergabe für die Regelung einer Lautstärke. Die Definition des Herstellers beschreibt einen Wertebereich von 0 -79 Dezimal. Wobei der Wert 0 die maximale Lautstärke definiert. Die Rechenregel erfolgt hierbei wie folgt:

Mit \# übergebener Wert: 10

-	10 -100	= -90

-	-90 *-1	= 90
-	90 / 1.2658	= 71,10128
-	71,10128 *1	= **71**

Das nomos system skaliert das Produkt einer Multiplikation auf die Nachkommastellen des Multiplikators. Ist der Multiplikator *1, also ohne Nachkommastellen. So ist auch das Produkt eine Ganzzahl.

- **[MAPPINGS](optional) Interpretation der Protokoll Rückgabesequenz**

  {MapString};{Bezeichnung};{Action};ONCHANGE;{Bedingung}

  {Map String} Eingangsstring, der unter Verwendung der Parser Optionen ausgewertet werden kann. Bei Verwendung ohne Parser ist die Bedingung erfüllt, wenn Eingangsstring = `{Map String}` . Bei Verwendung mit Parsern ist die Bedingung erfüllt, wenn ein erfolgreiches Match erfolgt. Der Inhalt der Parser Option wird als Argument übergeben.

  {Bezeichnung} Bezeichner für das Ergebnis des erfolgten Matches. Das Argument wird automatisch angefügt

  {Action} nomos Befehlskette und oder Script, welches bei erfolgtem Match ausgeführt werden soll.

  ONCHANGE Die Option `ONCHANGE` wird verwendet, wenn der Wert nur bei Wertänderung gesendet werden soll. Bleibt das Feld leer, wird der Wert immer gesendet.

  {Bedingung} Wenn definiert, wird vorherige Action nur ausgeführt, wenn Bedingung erfüllt. Es können Komperatoren („>", "<", "=") eingesetzt werden. Auch Kombinationen wie zB. „<>" (ungleich) oder „>=" (grösser oder gleich) können definiert werden. Durchsucht den Eingangsstring `{Map-String}` gemäß den nachfolgenden Bedingungen und übergibt die gefundene Zeichenkette (Match) als Argument an das Feld `{Bezeichnung}` . Hierbei kann in `{Bezeichnung}` auch ein frei definierbarer String als Klartext Bezeichnung eingetragen werden (s. Beispiel). In `{Action}` wird die nomos Befehlskette und/oder das Script eingetragen, welches zur Ausführung kommt, wenn die `{Bedingung}` erfüllt wurde.

Folgende Spezialzeichen können in `{Map String}` als Option für das Matchen oder Parsen verwendet werden:

Parser/Match Optionen

\^ Match't genau ein Zeichen und fügt es dem Argument an.

\? Match´t genau ein Zeichen und ignoriert es (Wildcard)

\^ und \? dürfen an beliebiger Stelle stehen

***** Wildcard-Match und fügt es dem Argument an

\% ignoriert beliebig viele Zeichen

\!{Anzahl}! überspringt {Anzahl} Stellen. {Anzahl} darf maximal 4 Ziffern haben und muss dezimal angegeben werden. Es ist zu beachten, dass bei einem HEX-Match ein Byte zwei Stellen entspricht.

Weitere Regeln: Kombinationen von `\^` und `\?` sind in beliebiger Länge möglich. Also Definitionen wie z.B. `\^^^^` für 4 Zeichen.

Wird `\^` und/oder `\*` kombiniert, wird das Argument entsprechend um die jeweiligen Zeichen ergänzt.

Die Einträge im `MAPPINGS` -Abschnitt werden gem. dem Schalter `MATCHING;NORMAL` (Reststring nach Match) oder `MATCHING;FULL` (gesamter Eingangsstring), zeilenweise abgearbeitet.

Bei einem Match `MATCHING;NORMAL` werden weitere Einträge ignoriert und die verbliebenen Zeichen in der Rückgabesequenz erneut evaluiert. Daher ist es sinnvoll, komplexe/lange Match-Strings an den Anfang des Mapping-Bereichs zu verschieben, um eine saubere Interpretation sicherzustellen. Im ASCII-Modus wird das Argument ohne weitere Wandlung angefügt; Im HEX-Modus wird das Argument in Dezimal umgewandelt und anschließend angefügt. Es können auch Hex-Werte (z.B. "0x0D", genau wie im COMMANDS-Bereich) vorkommen, auf die gematcht werden kann.

Beispiel (SERVERMODE;HEX):

07AA\##??;VOLUME 07EF\##04;CHANNEL Die Rückgabesequenz: `07EF040407AA1008`
erzeugt die folgende Klartext-Sequenz:
`<[CLASS]><CHANNEL=4><VOLUME=16></[CLASS]>`

Beispiel (SERVERMODE;ASCII):

TITLE=*|;TITLE Die Rückgabesequenz: `<ITUNES>TITLE=When the dog bells|ALBUM`
erzeugt die folgende Klartext-Sequenz:
`<[CLASS]><TITLE=When the dog bells></[CLASS]>`

Beispiel (SERVERMODE;ASCII) Erweitert:

<SYS>\%VOL=*|;VOLUME;<SYS><ENABLESS></SYS>;;<=5 Die Rückgabesequenz: `<SYS>GETVOL=EXT|VOL=4|...</SYS>`
erzeugt die folgende Klartext-Sequenz:
`<[CLASS]><VOLUME=4></[CLASS]>`
und löst gleichzeitig den Bildschirmschoner aus, da das Argument `<=5`

Beispiel (SERVERMODE;HTTP):

<head><title>standby</title></head><body>*</body></html>;STANDBY Die Rückgabesequenz:
`<head><title>standby</title></head><body>on</body></html>`
erzeugt die folgende Klartext-Sequenz:
`<[CLASS]>STANDBY=ON</[CLASS]>`

Sobald eine einzige `MAPPINGS` -Zeile existiert, wird nicht evaluierbarer Inhalt der Rückgabesequenz unterdrückt. So ist es möglich, nur die relevanten Daten aus einer Rückgabesequenz zu extrahieren, ohne Definitionen für alle möglichen Kombinationen anlegen zu müssen. Existiert kein MAPPINGS-Abschnitt, wird der raw-Output ausgegeben.

Die Kommunikation lässt sich im raw-Format (also nach der Umwandlung bzw. vor der Rückwandlung) jederzeit über den Logging-Monitor überwachen.

```
commandserver: UDP receive: 192.168.1.60:4003   data: \x0D\x0Astatus standby off pipoff recoff\x0D\x0A>
commandserver: parsing buffer: \x0D\x0Astatus standby off pipoff recoff\x0D\x0A>
commandserver: translated reply sequence: <TV_FITN>POWERMODE=standby|SUBMODE=off|PIPMODE=pipoff|RECMODE=recoff|OK</TV_FITN>
```

Für das weitere Verständnis erklären wir anhand einer einfachen Definitionstabelle den Aufbau einer Tabelle für einen LINN Verstärker.

Linn.csv

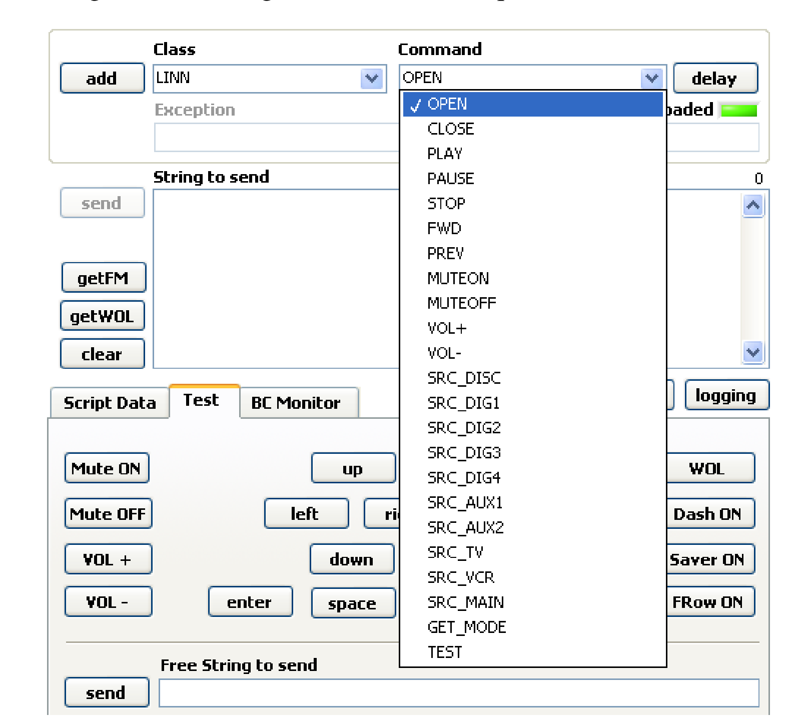
[CONFIG]	
SERVERIP	192.168.50.201
SERVERPORT	4001
SERVERMODE	ASCII
SERVERTIMEOUT	0
[COMMANDS]	
OPEN	$OPEN$\x0d\x0a
CLOSE	$CLOSE$\x0d\x0a
PLAY	$PLAY$\x0d\x0a
PAUSE	$PAUSE$\x0d\x0a
STOP	$STOP$\x0d\x0a
FWD	$SKIP +$\x0d\x0a
PREV	$SKIP -$\x0d\x0a
MUTEON	$MUTE\x20ON$\x0d\x0a
MUTEOFF	$MUTE\x20OFF$\x0d\x0a
VOL+	$VOLUME\x20+$\x0d\x0a
VOL-	$VOLUME\x20-$\x0d\x0a
SRC_DISC	$LISTEN\x20DISC$\x0d\x0a
SRC_DIG1	$LISTEN\x20DIG1$\x0d\x0a
SRC_AUX1	$LISTEN\x20AUX1$\x0d\x0a
SRC_AUX2	$LISTEN\x20AUX2$\x0d\x0a
SRC_TV	$LISTEN\x20TV$\x0d\x0a
SRC_VCR	$LISTEN\x20VCR$\x0d\x0a
SRC_MAIN	$LISTEN\x20MAIN$\x0d\x0a
GET_MODE	$MODE$\x0d\x0a

Wie vorher beschrieben, befindet sich im Abschnitt `[CONFIG]` die Einstellung für die Konfiguration der Kommunikationsparameter des zu steuernden Device. Da der LINN Unidisc über keine Netzwerkschnittstelle verfügt wurde hier ein MOXA LAN/RS232 Adapter eingesetzt. Der Moxa wurde mit der IP Adresse `192.168.50.201` belegt und der Port auf `4001` eingestellt. Das LINN Protokoll ist ein reines ASCII Protokoll. Ein Feedback wird nicht ausgewertet `(TIMEOUT=0)` . Das Fehlen der Einstellungen `SERVERPROTOCOL` hat zur Folge, das dass UDP Protokoll (Standard) eingestellt wurde. Da dieses Protokoll keine Regel zur Berechnung der Checksumme enthält, fehlt auch der Eintrag `SERVERTYPE` .

Unter dem Bereich `[COMMANDS]` sind nun die eigentlichen Befehle eingetragen. Gem. Hersteller- dokumentation muss dem LINN folgende Befehlskette zum Öffnen des Laufwerkschachtes übertragen werden `$OPEN$\x0d\x0a` Dieser Befehl soll zukünftig im nomos Protokoll unter `<OPEN>` zur Verfügung gestellt werden. Ebenso wird mit allen weiteren notwendigen Befehlen verfahren.

Ist die Tabelle erstellt und in dem entsprechenden Verzeichnis auf Ihrem nomos system abgelegt, muss der nomos Service neu gestartet werden bzw. ein „RESET“ über den nomos scripting Client ausgelöst werden. Danach führen Sie im Dateidialog den Befehl „AddOn Commands laden“ aus. Wählen Sie nun die Klasse „`LINN`“ , die aufgrund des Dateinamens der Definitionstabelle definiert wurde, aus und betrachten Sie die neuen Befehle. Anstelle des nomos scripting Client können Sie die Befehlssequenz auch mit einem einfachen Terminal Programm (zB NetCAT oder Hyperterminal) senden. Auch

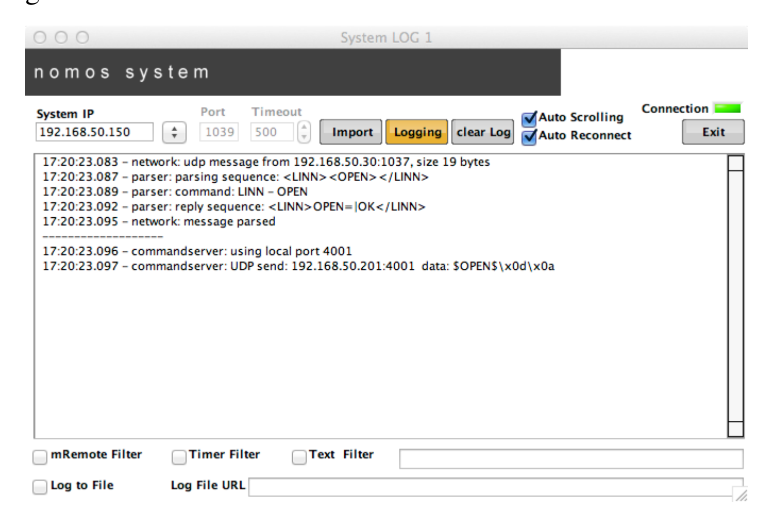
der nomos Wizard verfügt über die Möglichkeit, Befehlssequenzen manuell zu versenden.



Sie sehen zB im scripting Client, dass nun alle Befehle, die Bestandteil der `CommandServer` - Definitionstabellen sind, zur Verfügung gestellt werden. Wählen Sie nun den „open“ aus und fügen Sie diesen in die „String to send“ Maske ein.



Senden Sie den Befehl an Ihr nomos system und beobachten Sie dabei den Eintrag in dem Logging Monitor des Scripting Client.



Sie sehen, dass nomos die Daten entsprechend den Einstellungen der Definitionstabelle behandelt und das notwendige Datenpaket an die eingestellte Zieladresse sendet.

Weitere Möglichkeiten der `[MAPPINGS]` Definitionen möchten wir nachfolgend erläutern:

data: \x0D\x0Astatus tv off pipoff recoff\x0D\x0A> Empfangssequenz, die nach dem Einschalten des Devices empfangen wird

[MAPPINGS]

status * ;POWERMODE;<SCRIPT><RUN=LOEWE_MEDIEN_\#.myh></SCRIPT>;NOCACHE
Matcht auf die Zeichenkette „`tv`" und übergibt den Inhalt als Wert für `POWERMODE` `(POWERMODE=tv)` . Da ein Match erfolgt, also die Match Bedingung erfüllt wurde, wird die local Action ausgeführt. In der local Action wird ein Script ausgeführt. Der Scriptname wird nun durch den Inhalt des Match erweitert zu `LOEWE_MEDIEN_tv.myh` und entsprechend ausgeführt. Die Ausgabe erfolgt bei jedem empfangenen Event (`NOCACHE`).

status \% * ;SUBMODE Match auf die Zeichenkette, die zwischen dem 2. und 3. Leerzeichen nach „`status`" empfangen wird. In diesem Fall ist das der Wert „off". Dieser Wert wird als `SUBMODE` bezeichnet und entsprechend ausgegeben `SUBMODE=off` . Die Ausgabe erfolgt nur bei einer Wertänderung. (`NOCACHE` nicht gesetzt)

status \% \% * ;PIPMODE Wie vor, matcht jedoch auf die Zeichenkette, die zwischen dem 3. und 4. Leerzeichen nach „`status`" und belegt `PIPMODE` mit dem entsprechenden Inhalt `PIPMODE=pipoff`

status \% \% \% *\0x0D;RECMODE Wie vor, matcht jedoch auf die Zeichenkette, die zwischen dem 4. Leerzeichen und dem `\0x0D` nach „`status`" und belegt `RECMODE` mit dem entsprechenden Inhalt `RECMODE=recoff`

Ist die Protokollsequenz vollständig durchlaufen bzw. ausgewertet, wird die übersetzte Meldung ausgegeben: `<TV_FITN>POWERMODE=tv|SUBMODE=off|PIPMODE=pipoff|RECMODE=recoff|OK</TV_FITN>`

Das nomos system kann empfangene Sequenzen von Fremdsystemen auswerten und bei Bedarf weitere Aktionen antriggern. Hierbei können empfangene Daten ausgewertet, umgewandelt und an die auszuführende Aktion übergeben werden. Wir nennen dies eventbasierte Folgesteuerung.

data volume *\0x0D;VOLUME;<AVR_GAL><VOLUME_SET=\#></AVR_GAL>;
Empfang der aktuellen Lautstärke des TV Systems mit anschließender Weiterleitung an den AVR. Hierüber kann beispielsweise die Lautstärke Regelung eines angeschlossenen AV Receivers über die Fernbedienung des TV-Systems im Hintergrund erfolgen.

4.1.2 Reservierte (extended) Servertypen `[SERVERTYPE]`

Wie vorher beschrieben, gibt es Serverklassen bzw. deren Abhandlung der Checksummen Berechnung, die bereits fest in nomos implementiert wurden. Diese speziellen Servertypen können nur vom nomos Entwicklerteam eingebunden werden . Hierüber ist es beispielsweise möglich, Checksummen Berechnungen o.ä. zu hinterlegen. Generell werden derartige Protokolle komplett vom nomos Entwicklerteam eingebunden und die entsprechenden Definitionstabellen, sofern vorhanden, frei zur Verfügung gestellt.

Selbstverständlich können entsprechende Protokolle auch über die vorher beschriebenen möglichen Verfahren eingebunden werden, jedoch kann bei Protokollen mit z.B. anhängiger Checksumme kein dynamischer Wert übergeben werden. Dies würde z.B. bei einem Multiroom-System bedeuten, dass je Befehl eine Zeile in der Definitionstabelle eingefügt werden muss.

Zurzeit sind folgende extended Servertypen eingebunden:

- **RUS** ermittelt und sendet die Checksumme für das Russound Protokoll (HEX)
- **ADNOTAM** ermittelt und sendet die Checksumme für das Adnotam Protokoll (HEX)
- **SPEAKERCRAFT** ermittelt und sendet die Checksumme für das Speakercraft Protokoll (HEX)

Für die Anwendung eigener Checksummen Regeln steht die Option:

- **SUM** variable Checksummen Option

zur Verfügung.

Eine genaue Erklärung zur Anwendung finden Sie im vorherigen Kapitel CommandServer.

4.1.3 Verwendung von USB/RS232 Umsetzern

Das nomos system unterstützt die Verwendung lokaler USB/RS232-Adapter/Umsetzer. Hierbei kann auf externe IP/RS232-Umsetzer verzichtet werden, sofern Ihre Installation dies zulässt. Nachfolgend wird die Vorgehensweise zur Verwendung dieser Adapter beschrieben bzw. die notwendigen Einstellungen in der `.csv` Datei verdeutlicht.

Anwendung:

SERVERIP;LOCAL Betriebsart für lokale Devices

SERVERPORT AUTO oder
{Serieller Port{Geschwindigkeit};{Datenbits};{Parität};{Stopbits}{Flußkontrolle}

Erklärung der Variablen `SERVERPORT` **bei** `SERVERIP=LOCAL` **:**

Bei Verwendung von USB/RS232 Adapter muss die Einstellung **SERVERIP=LOCAL** erfolgen.

AUTO Sucht den 1. seriellen Port im System und nutzt ihn automatisch. Die Nutzung macht eigentlich nur dann Sinn, wenn man nur einen USB/RS232 Adapter installiert hat.

{Serieller Port} der Schnittstellenname im Format `/dev/cu.xxxxxxx`

Die Verwendung des Befehls `GETSERIALPORTLIST` aus der `<SYS>` Klasse gibt Auskunft über die vom System verwendeten Schnittstellen- bzw. Devicenamen.

{Geschwindigkeit} die Geschwindigkeit, mit der der serielle Port betrieben werden soll, getestet wurden `9600`, `19200` und `115200 bps`

{Datenbits} Anzahl der Datenbits (5-8)

{Parität} 'n' -> keine
'e' -> gerade
'o' -> ungerade

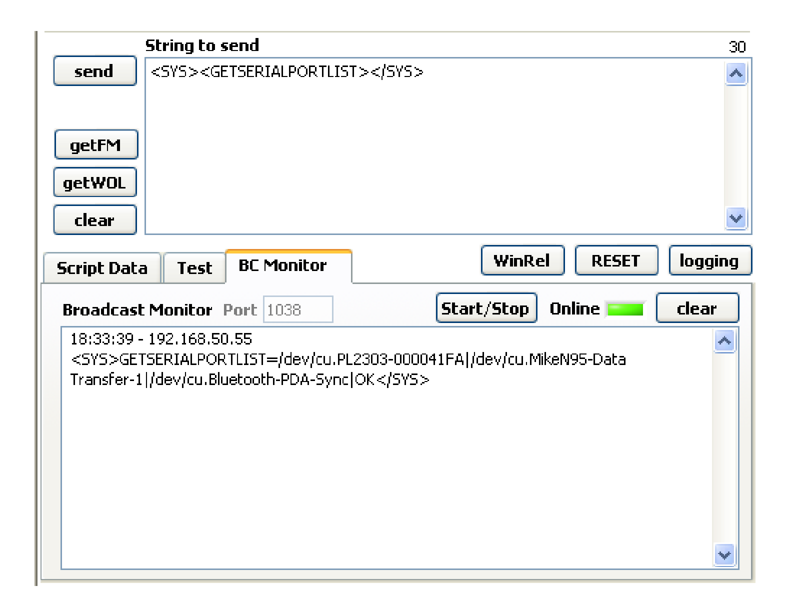
{Stopbits} Anzahl der Stopbits (`1-2`)

{Flusskontrolle} `NONE` , `RTSCTS` , `DTRDSR` , `XONXOFF`

Beispiel Einstellung für USB/RS232 Adapter:

SERVERIP;LOCALSERVERPORT;AUTO;115200;8;n;1;RTSCTSSERVERPROTOCOL;SERIAL
Sucht und öffnet den 1. gefundenen lokalen USB/RS232-Adapter und stellt eine COM- Schnittstelle mit den Parametern `115200,8,n,1` und aktivierter `RTSCTS` Flusskontrolle bereit.

Um Auskunft über die teilweise doch kryptischen Device-Namen zu erhalten, wurde der Befehl `GETSERIALPORTLIST` aus der Klasse `<SYS>` (Systembefehle) eingebunden. Verwenden Sie einfach den Scripting Client um an die entsprechenden Namen zu gelangen.



Alle Rückmeldungen mit dem Inhalt:

<SYS>GETSERIALPORTLIST=/dev/cu.PL2303-0000201A| sind entsprechende USB/RS232-Device-Treiber. In gezeigtem Fall wäre der korrekte Eintrag in der `.csv` Datei:

SERVERPORT;/dev/cu.PL2303-0000201A;115200;8;n;1;RTSCTS Entsprechend zu verwenden, wenn ein Device unter seiner direkten Adresse angesprochen werden soll bzw. wenn mehr als ein USB/RS232-Device an dem System verwendet wird.

Die Erste in der zurück gelieferten Liste auftauchende Bezeichnung entspricht der Auto-Funktion in der `.csv`

Zum Einrichten eines Systems ist es sinnvoll, nach und nach die Konverter anzuschließen und zwischenzeitlich immer wieder den o.g. Befehl aufzurufen, um die richtige Zuordnung und den entsprechenden Portnamen zu bekommen. Die Portbezeichnung richtet sich danach, in welchen USB-Port der Konverter gesteckt wird. Das kann natürlich auch an einem externen USB- Hub erfolgen.

Wir haben Adapter mit dem Profilic PL- 2303 Chipsatz erfolgreich testen können und empfehlen dementsprechend auch ausschließlich deren Verwendung.
Den entsprechenden Apple OSX-Treiberdownload für den Prolific PL2303 (der Chipsatz wird von vielen verschiedenen Herstellern verwendet) finden Sie unter nachfolgender Adresse.

http://sourceforge.net/projects/osx-pl2303/

Ebenso befindet sich der Treiber auf der beigefügten CD unter dem Verzeichnis `\Tools und Treiber\Profilic`.

Bitte beachten Sie, dass die Verwendung des Treibers der GNU General Public License (GPL) unterliegt.

4.2 GIRA Homserver KO-Gateway, KNXnet/IP Support

Im Nachfolgenden beschreiben wir die möglichen Schnittstellen für die EIB/KNX Anbindung. Diese Schnittstellen werden wir entsprechend der Markterfordernisse permanent anpassen und erweitern.

4.2.1 KNX-IP BAOS 770

nomos unterstützt das BAOS (Bus Access and ObjektServer) Protokoll und ermöglicht somit die direkte Verarbeitung von bis zu 256 KNX-Objekten je nomos System. Es ist dafür gedacht, einfache KNX-Abläufe wie z.B. das Antriggern von Lichtszenen oder einfache Lichtschaltungen über Buttonleisten oder aber die mremote Visualisierung einzubinden. Ebenso können hierüber Aktionen auf dem nomos system (Scriptaufruf oder Befehlsabläufe) getriggert werden. Um diese Schnittstelle nutzen zu können, ist das Weinzierl KNX IP BAOS 770 Interface zwingend notwendig. Die Konfiguration und die Definition der KNX-Datenpunkte wird im BAOS-Interface über die ETS (EIB-Tool-Software) vorgenommen. Die Vorgehensweise entnehmen Sie der Dokumentation des Herstellers.

Ist das Interface entsprechend eingerichtet, kann nomos mit dem Gerät kommunizieren und Daten austauschen.

Durch die Einbindung des KNXnet/IP-Protokoll für Routing und Tunneling wird das BAOS-Objekt/Serverkonzept nicht weiter verfolgt werden. Das Weinzierl KNX-IP BAOS770-Interface kann auch im KNXnet/IP Tunneling Modus betrieben werden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	baos.csv
Länge:	1000 Zeilen/Einträge
Event:	BC, Status
Export:	nein
Lizenz:	ja

Die `baos.csv` **besteht aus 3 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DATAPOINTS]; Definition der Datenpunkte, die zus. auf den BC Port übertragen werden sollen (optional)

[ACTIONS]; Definition der Datenpunkte, die eine direkte Aktion (Script Aufruf) auslösen sollen

Aufbau:

- **[CONFIG]; Sektion**

  BAOSIP;{IP}/OFF/AUTO;{IP} IP des KNXBAOS (fest);

  OFF, BAOS-Funktion abgeschaltet;

  AUTO, IP des BAOS wird autom. im aktuellen Subnet gesucht

- **[DATAPOINTS]; Sektion** (optional, für den Empfang oder Script Aufruf notwendig):

  {Nummer};{Name};{Typ} Aufbau der Definition eines digitalen Joins

Erläuterungen:

{Nummer} BAOS Objekt Nummer/Datenpunkt

{Name} Frei definierbarer Klartextname für das Objekt

{HEX/DEC/ASCII} Typ/Darstellungsmodus des Objekts

- **[ACTIONS]; Sektion**

  {Nummer};{Wert}/{#};{Action};{Typ};

Erläuterungen:

{Nummer} BAOS Objekt Nummer/Datenpunkt

{Wert}/{#} Wert wann Aktion ausgeführt werden soll. # = alle Werte

{Action} nomos Sequenz oder Script, dass ausgeführt werden soll, wenn die Bedingung `{Wert}` wahr ist. Wird der Platzhalter # verwendet, kann der Wert mittels \# an das Script oder die nomos Sequenz übergeben werden

{Typ} Typ/Darstellungsmodus des Objekts `HEX(default)/DEZ/ASCII`

Beispiele für die Definitionen [DATAPOINTS] Sektion:

9;Klingel;HEX generiert die Broadcast-Ausgabe: <BAOS>Klingel='VALUE'</BAOS>

Beispiele für die Definitionen [ACTIONS] Sektion:

9;0A;script.myh;HEX Führt das Script `script.myh` aus, wenn Datenpunkt 9 den Wert 10 hat

9;10;script.myh;DEC exakt wie vor beschrieben

9;ON;script.myh;ASCII Führt das Script `script.myh` aus, wenn Datenpunkt 9 der ASCII-Sequenz "ON" entspricht

9;#;script.myh;DEC Führt das Script `script.myh` aus, wenn Datenpunkt 9 den Wert ändert und übergibt den Wert dezimal als `[ARG]` an das Script.

9;#;script.myh;HEX Führt das Script `script.myh` aus, wenn Datenpunkt 9 den Wert ändert und übergibt den Wert als `[ARG]` hexadezimal an das Script

9;#;script.myh;ASCII Führt das Script `script.myh` aus, wenn Datenpunkt 9 den Wert ändert und übergibt die ASCII-Sequenz als `[ARG]` an das Script

Sobald Datenpunkte definiert sind, baut der Daemon eine feste, dauerhafte Verbindung zum `BAOS` auf und gibt empfangene Wertänderungen der in der `.csv` aufgelisteten Datenpunkte sofort über den Broadcast-Port aus. Über den `Eventserver` kann dann über eine entsprechende Definition eine entsprechende Aktion ausgelöst werden.

Wichtige Hinweise:

Generell werden die Datenpunkte immer über ihre Objeknummer angesprochen, also nicht über die entsprechende EIB/KNX Gruppenadresse, sondern über das Objekt mit dem die Adresse assoziiert wurde. Anstelle des Aufrufs einer Script Datei können auch direkte Befehlsketten (Kommandosequenzen) verwendet werden.

Beispw.: `9;1;<SYS><MUTON></SYS>;DEC`

Es existiert eine Befehlsklasse `BAOS` mit folgenden Befehlen:

Kommando	Argument	Beschreibung
GETINFO		liefert allgemeine Informationen über den BAOS
GETDESCRIPTION=	{Datenpunktnummer}	liefert die Einstellungen des gewählten Datenpunktes
GETDESCRIPTIONSTRING=	{Datenpunktnummer}	liefert die Beschreibung des gewählten Datenpunktes
GETVALUE=	{Datenpunktnummer}	liefert den aktuellen Wert des Datenpunktes inkl. Zustandsflags im HEX-Format
SETVALUE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	setzt den Wert des gewählten Datenpunkt in HEX, Dezimal oder als ASCII
SENDVALUE=	{DatenpunktNr.}, { HEX/DEC/ASCII}	sendet den Wert auf den KNX-Bus
SETANDSENDVALUE=	{DatenpunktNr.}, { HEX/DEC/ASCII}	Kombination von `SETVALUE` und `SENDVALUE`
READVALUE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	liest den Wert vom KNX-Bus
CLEARTRANSMISSIONSTATE=	{DatenpunktNr.}, {HEX/DEC/ASCII}	löscht den Verbindungsstatus
GETPARAMETERBYTE=	{ParameterNr.}	liefert das Byte des gewählten Parameters

Beispiele:

<BAOS><SETANDSENDVALUE=4,1,DEC></BAOS> Setzt das BAOS Objekt 4 auf den Wert 1

<BAOS><GETINFO></BAOS> Frag die Parameter des BAOS Interface ab. Die Ausgabe erfolgt auf dem BC Port.

Wichtige Hinweise:

Wenn als Typ `ASCII` angegeben wird, wird der String auf `14 Bytes` erweitert bzw. abgeschnitten.

Wird kein Argument angegeben, wird von einem Dezimal Wert ausgegangen. Bei dem Typ „`HEX`“ ist „`0x`“ als Prefix nicht notwendig.

4.2.2 GIRA Homeserver KO-Gateway Support

Das nomos system unterstützt den bi-direktionalen Zugriff auf das GIRA Homserver KO-Gateway. Diese ermöglicht die einfache Datenübergabe, da der Austausch von Informationen bis hin zum Zugriff auf den EIB/KNX direkt über das Gateway erfolgen kann. Die Kommunikation zwischen nomos und dem GIRA Homeserver muss dann nicht mehr zwingend über UDP/TCP Kommunikation erfolgen.

Wichtige Hinweise:

Bitte beachten Sie, dass die Datenpunkte, die Sie auswerten/ansprechen möchten im GIRA Homeserver für die Kommunikation über das KO-Gateway freigegeben wurden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	hs.csv
Länge:	10000 Zeilen/Einträge
Event:	BC, Status
Export:	nein
Lizenz:	ja

Die `hs.csv` **besteht aus 2 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DATAPOINTS]; Definition von Datenpunkten, die über das GIRA Homeserver KO-Gateway empfangen und ausgewertet werden sollen.

Aufbau: Im misc-Ordner eine `hs.csv` mit folgendem Inhalt anlegen:

- **[CONFIG];Sektion**

  ACTIVE;YES; Aktiviert den KO-Gateway Support
  „YES“ (Default) / NO

  DEBUG;YES Erweitertes Protokoll aktiv
  “YES”/”NO” (Default)

  HSIP;{IP Adresse} IP Adresse des GIRA Homservers

  HSPORT;{KO-Portnummer} Port des GIRA KO-Gateway

  HSKEY;{ein evtl. im HS festgelegtes Passwort} Passwort/Key des KO-Gateway

  ADDRTYPE;{Adresstyp} Legt fest, wie die Adressausgabe im Broadcast erfolgen soll. Mögliche Werte „2STEP“ oder 3STEP“. Leer oder nicht definiert = Dezimalausgabe.

- **[DATAPOINTS]; Sektion**

  Optional / zusätzlich (für den Empfang oder Script Aufruf notwendig)

  {KO im GIRA Homeserver};{Datenpunkt-Name};{Bedingung};{mmh-Sequenz }

  Erläuterungen:

  {KO im GIRA Homeserver} Adresse des Kommunikationsobjekts

  {Datenpunkt-Name} darf leer sein und ist frei definierbar

{Bedingung} kann (entsprechend BAOS) folgende Zustände haben:

{Match-String} der vom HS übermittelte Wert muß mit `{Match-String}` übereinstimmen

{#} alle Werte starten das Script bzw. die Sequenz wenn `{Bedingung}` leer ist, wird `'#'` angenommen.

{mmh-Sequenz} Auszuführende Kommandosequenz oder Script, wenn Bedingung erfüllt. Kann als Platzhalter für den Wert '\#' beinhalten, bei Scripten wird der empfangene Wert als Argument übergeben.

Wichtige Hinweise:

Bitte beachten Sie, dass das KO-Gateway Werte, die keinen ASCII Text beschreiben, als Fließkomma Wert überträgt. So wird ein binäres „`1`" oder „`0`" als „`1.0`" oder „`0.0`" übergeben. Entsprechend ist das Feld `{Bedingung}` zu definieren.

Beispiele für die Definitionen [DATAPOINTS] Sektion:

8/1/4;iTunes Next;1.0;<ITUNES><NEXT><PLAY></ITUNES> Bei Empfang der Adresse 8/1/4 mit dem Wert „1" führt iTunes den internen Befehl „`NEXT`" und `PLAY` aus. Auf dem Statusport wird die Meldung „`iTunes Next=1.0`" generiert.

8/1/7;TV ON;1.0;<SCRIPT><RUN=StartTV.myh></SCRIPT> Bei Empfang der Adresse 8/1/7 mit dem Wert „1" wird das Script „`StartTV.myh`" ausgeführt. Auf dem Statusport wird die Meldung „`TV ON=1.0`" generiert.

5/1/0;Playlist;#;<SCRIPT><RUN=Play(Playlist).myh></SCRIPT> oder, gleichwertig

5/1/0;Playlist;#;<ITUNES><PLAYPLAYLIST=\#></ITUNES> Bei Empfang der Adresse 5/1/0 (EIS15 ASCII Definition) mit dem angenommenen Wert „`Musik`" wird das Script „`Play(Playlist).myh`" ausgeführt und der eingehende Text als Argument übergeben. Auf dem Statusport wird die Meldung „`Playlist=Musik`" generiert.

Es existiert eine Befehlsklasse HS mit folgenden Befehlen:

Kommando	Argument	Beschreibung
SETVALUE#	{Adresse},{Wert}	setzt {Wert} an {Adresse} absolut.
SETVALUEREL#	{Adresse},{Wert}	setzt {Wert} an {Adresse} relativ.
STEPUP#	{Adresse}	erhöht den Wert an {Adresse}.
STEPDOWN#	{Adresse}	senkt den Wert an {Adresse}.
LISTUP#	{Adresse}	erhöht den Wert an {Adresse}.
LISTDOWN#	{Adresse}	senkt den Wert an {Adresse}.

Statt `{Adresse}` kann bei allen Befehlen auch der in der `.csv` eingetragene Datenpunkt-Name ver-

wendet werden. Wenn der HomeServer innerhalb von 3 Sekunden auf den Befehl antwortet, wird der empfangene Wert als Rückgabewert ausgegeben. Außerdem werden alle vom HS empfangenen Werte über den Broadcast-Port gesendet:

<HS><{Adresse bzw. Name, wenn definiert}={Wert}><.....></HS>

Beispiele:

<HS><SETVALUE=4/1/1,1></HS> Setzt die Adresse `4/1/1` auf den Wert `1`

<HS><LISTUP=110/1/1></HS> Setzt den Wert der internen Adresse entsprechend der voreingestellten Liste auf den nächsten Wert.

4.2.3 KNXnet/IP (Routing und Tunneling)

KNXnet IP Routing-Support
KNXnet IP Tunneling Support
KNXnet IP TUNNELING_BRIDGE Support

Das nomos system unterstützt auch das KNXnet/IP Protokoll. Diese Einbindung ist die z.Z. optimalste Schnittstelle zum KNX/EIB, da es quasi keine Einschränkung in der Anzahl der Datenpunkte zugrunde legt und keine zusätzliche Parametrierung innerhalb der ETS, wie es zum Beispiel bei dem BAOS Objekt Server notwendig ist, vorgenommen werden muss.

Es werden beide Protokolltypen, also KNXnet/IP Routing und KNXnet/IP Tunneling unterstützt. Ebenso ist es möglich, eine „`TUNNELING_BRIDGE`" einzurichten. Dieser Modus wird im nachfolgenden noch näher erläutert.

Die Datenpunktdefinitionen werden in einer gesonderten Datei (`knx.esf`) im Ordner `misc` abgelegt. Diese Datei unterliegt dem OPC Export Format der ETS und kann direkt aus der ETS heraus exportiert werden. Es werden alle Datenpunkttypen unterstützt. Ebenso werden die dort definierten Namen für die Broadcast-Ausgabe und im Logging-Monitor verwendet. Die Typenkonvertierung und Skalierung erfolgt somit automatisch und muss nicht weiter berücksichtigt werden. Es kann auch ein individueller Name für die `.esf` -Datei vergeben werden. Dies ist in der `CONFIG` -Sektion der `knx.csv` einstellbar.

Bei unbekannten Datentypen in der `.esf` -Datei erfolgt ein Log-Eintrag und die entsprechende Definition wird ignoriert. Datenpunkte, die nicht in der `.esf` -Datei eingetragen sind, werden ignoriert.

Folgende Datenpunkttypen werden zZ unterstützt:

EIS 1	'Switching' (1 Bit)
EIS 2	'Dimming - position' (1 Bit)
EIS 2	'Dimming - control' (4 Bit)
EIS 2	'Dimming - value' (8 Bit)
EIS 3	'Time' (3 Byte)
EIS 4	'Date' (3 Byte)
EIS 5	'Value' (2 Byte)
EIS 6	'Scaling - percent' (8 Bit)
EIS 6	'Scaling - degree' (8 Bit)
EIS 7	'Drive control' (1 Bit)
EIS 8	'Priority - position' (1 Bit)
EIS 8	'Priority - control' (2 Bit)
EIS 9	'Float value' (4 Byte)
EIS 10	'16Bit Counter' (2 Byte)
EIS 11	'32Bit Counter' (4 Byte)
EIS 12	'Access' (4 Byte)
EIS 13	'EIB-ASCII-Char' (8 Bit)
EIS 14	'8Bit Counter' (8 Bit)
EIS 15	'Character String' (14 Byte)

Da die OPC Exportschnittstelle fehlerbehaftet ist, (`2Byte` float-Werte, wie z.B. Temperaturen werden als „`uncertain`" = undefiniert, übergeben) haben wir eine Anpassung vorgenommen. So werden alle `2Byte uncertain` Datenpunkte automatisch als „`EIS5'Value'(2Byte)`" interpretiert.

KNXnet/IP Routing-Support

Allgemeines:

Beim Start wird automatisch ein KNX/IP routingfähiger IP-Router gesucht und gegebenenfalls im Log angezeigt. nomos liest die Mutlicast-Adresse des IP-Routers aus und tritt der entsprechenden Gruppe bei.

Vorteil der Routing Variante:

Es können beliebig viele nomos systeme gleichzeitig auf das KNXnet/IP-Device gleichberechtigt zugreifen. Die Konfigurationsdatei heisst knx.csv und wird im misc-Ordner abgelegt. In der knx.csv angelegte Adressen werden konvertiert, ggf. Actions ausgeführt und über den Broadcast Port ausgegeben.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	knx.csv
Länge:	10000 Zeilen/Einträge
Event:	BC, Status
Export:	nein
Lizenz:	ja

Die `knx.csv` besteht aus 3 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DATAPOINTS]; Definition von Datenpunkten, die über KNXnet/IP empfangen und ausgewertet werden sollen.

[TABLE={NAME}]; Scan Definitionstabellen, Gruppenpollen

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES; Schaltet den KNXnet/IP Support
  „`YES`" = ein,
  „`NO`" = aus

  DEBUG;YES; Schalter für
  „`YES`" = erweiterte Logausgaben (Busmonitor)
  „`NO`" = standard Logausgaben

  ESF;{ESF File}; Name der `.esf` -Datei in dem Ordner misc (`knx.esf` = Default)

  PHYSADDR;{Phys.Adresse}; Lokale Physikalische Absenderadresse, die mmh beim Versenden verwenden soll.

  ADDRTYPE;{Adresstyp}; „`2STEP`" = 2stufige Darstellung,
  „`3STEP`" = 3stufige Darstellung (Default)

  DPTNAMES;YES; Schaltet die Namen der Datenpunkte im Broadcast
  „`YES`" = ein (Default),
  „`NO`" = aus

  MATCHLOCAL;YES; Legt fest, ob die unter `DATAPOINTS` definierten Aktionen auch dann ausgeführt werden, wenn der Daemon selbst auf die Datenpunkte

schreibt.
„YES" = ein,
„NO" = aus (Default)

INITSCAN;YES; scannt die definierten Tabellen beim Engine-Start
„YES/ALL" = ein,
„NO" = aus (Default)

SENDRATE;{Telegr/s}; Begrenzt die Anzahl der Telegramme die pro Sekunde auf den Bus beschrieben werden. Der Puffer ist variabel und Burst fähig. „17" (Default). Der hier eingestellte Wert gilt auch für den Abstand der Telegrammabfragen, die mit SCAN= initiert werden.

CONNECTIONTYPE;{Protokolltyp}; „ROUTING" (default: ROUTING) oder
„TUNNELING" oder
„TUNNELING_BRIDGE"

Spezifische Einstellungen für die Betriebsart „ROUTING“:

CONNECTIONTYPE;ROUTING;

Bei CONNECTIONTYPE;ROUTING wird folgender, zusätzlicher Schalter benötigt:

MULTICAST_IP;{Multicast ADR}; Multicast IP des Gerätes Standard: {224.0.23.12} oder
„AUTO" für automatisches Suchen
(Default: AUTO)

- **[DATAPOINTS];Sektion**

  {KNX/EIB Gruppenadresse};{Bedingung};{mmh-Sequenz}; In diesem Bereich können KNX Events direkt ausgewertet und entsprechende Aktionen eingeleitet werden.

  Erläuterungen:

  {KNX/EIB Gruppenadresse} Gruppenandresse im entsprechend eingestelltem ADDRTYPE

  {Bedingung} kann (entsprechend BAOS) folgende Zustände haben:

  {Match-String} der von KNX/IP übermittelte Wert muß mit {Match-String} übereinstimmen.

  {#} alle Werte starten das Script bzw. die Sequenz. Wenn {Bedingung} leer ist, wird '#' angenommen.

  {mmh-Sequenz} Auszuführende Kommandosequenz oder Scriptname, wenn Bedingung erfüllt. Kann als Platzhalter für den Wert \# beinhalten, bei Scripten wird der empfangene Wert als Argument übergeben

- **[TABLE={NAME}];Sektion**

  {NAME} {NAME} definiert eine entsprechende Gruppe. Es können beliebig viele [TABLE={name}] Sektionen angelegt werden. Diese Namen werden bei der Ausführung der SCAN Befehle benötigt und im weiterem Verlauf genauer erklärt.

{KNX/EIB Gruppenadresse}; Gruppenadresse die gescannt werden soll. Es darf nur eine Gruppenadresse je Zeile eingetragen werden.

Beispiele für die Definitionen [DATAPOINTS] Sektion:

8/1/4;1;<ITUNES><NEXT><PLAY></ITUNES> Bei Empfang der Adresse 8/1/4 mit dem Wert „1" führt iTunes den internen Befehl „`NEXT`" und `PLAY` aus.

Pro Adresse lassen sich mehrere `Actions` definieren, wenn unterschiedliche `Match` -Bedingungen angegeben werden. Bei identischen `Match` -Bedingungen pro Adresse wird nur die erste gefundene `Action` ausgeführt.

15/7/10;1;<SYS><SAY=on></SYS> oder 15/7/10;0;<SYS><SAY=off></SYS> Führt nur bei Empfang einer logischen „1" der Adresse 15/7/10 den Befehl `<SYS><SAY=on></SYS>` aus. Bei empfang einer logischen „0" wird nur der Befehl `<SYS><SAY=off></SYS>` ausgeführt.

8/1/7;#;<SYS><VOLSET=\#></SYS>; Schreibt den empfangenen Wert auf die System Volume.

5/2/8;DOWN,100;<SYS><VOLDN=5></SYS>;5/2/8;UP,100;<SYS><VOLUP=5></SYS>; Empfängt und wertet ein 4Bit Dimmtelegramm (EIS2) aus. Hierbei empfiehlt es sich, dass entsprechende KNX Telegramm zyklisch senden zu lassen (Einstellung am entspr. Sensor beachten), da der entsprechend auszuführende Befehl `{mmh-Sequenz}` nur je empfangenem Telegramm angetriggert wird.

Beispiele für den Scan Support:

```
[TABLE=Wohnzimmer];
1/8/4
1/8/5
1/8/7
1/4/3

[TABLE=Schlafzimmer];
1/4/3
1/3/5
1/2/7
```

Definiert zwei Scan Tabellen, die unter Verwendung der SCAN Befehle abgerufen werden können. Der SCAN kann unmittelbar erfolgen, oder aber im Hintergrund ablaufen. Bitte beachten, dass ein SCAN nur funktionieren kann, wenn auch entsprechend das „l" Flag des assoziierenden KNX Kommunikationsobjekt gesetzt ist. Je Adresse sollte dieses Flag nur einmalig an einem Kommunikationsobjekt gesetzt sein.

Die Unterscheidung in den beiden verschiedenen SCAN Methoden liegt im zeitlichen Abstand der Leseanforderungen. Mit `SCAN=` können schnelle Abfragen generiert werden. Hier sollte jedoch beachtet

werden, dass nicht zu viele Telegramme mit dieser Geschwindigkeit abgefragt werden. Für die störungsfreie Abfrage vieler Telegramme, wie zb für einen initial Scan, ist der `BACKGROUNDSCAN=` vorgesehen.

Die Telegramme werden sequentiell nach Erhalt einer Antwort ausgeführt. Auf eine Antwort wird max. 1s gewartet. Wird innerhalb dieser Zeit keine Antwort empfangen, wir die Meldung `ERR_NO_RESPONSE` generiert. Die Antworten des Scan's erscheinen ebenfalls im Broadcast (BC):

bc: <KNX><15/2/181-Geli.DimBelLlp.ein/ausStatus=0></KNX>

bc: <KNX><15/5/28-SOLL_TEMP_Serverschrank=27.00></KNX>

bc: <KNX><15/2/21- mike.DimBelLlp.ein/ausStatus=0></KNX>

Es existiert eine Befehlsklasse KNX mit folgenden Befehlen:

Kommando	Argument	Beschreibung
SETVALUE=	{Adresse},{Wert}	setzt `{Wert}` an `{Adresse}`
GETVALUE=	{Adresse}	Liest einen Wert einer `{Adresse}`
SCAN=	{Tabellenname}	Scannt eine Liste von Adressen wie in der `knx.csv` definiert
BACKGROUNDSCAN=	{Tabellenname},{ALL}	Wie vor jedoch erfolgt der Scan im Hintergrund bei `{ALL}` werden alle eingetragenen Tabellen gescannt

Beispiele:

<KNX><SCAN=Wohnzimmer></KNX><KNX><SCAN=Wohnzimmer><SCAN=Schlafzimmer></KNX>
Löst die Abfrage der Gruppenadressen, wie z.B. unter `[TABLE=Schlafzimmer]` definiert aus. Es können auch mehrere Tabellen gleichzeitig abgefragt werden.

<KNX><BACKGROUNDSCAN=Schlafzimmer></KNX> Löst den Hintergrundscan der Tabelle Schlafzimmer aus. Ein Hintergrundscan wird fix mit ca. 3 Telegramme/s ausgeführt.

<KNX><SETVALUE=1/2/3,1></KNX> Setzt den Wert der Gruppenadresse `1/2/3` auf `1`

<KNX><SETVALUE=0/0/1,[TIME]></KNX> Setzt den Wert der Gruppenadresse `0/0/1` auf die aktuelle Systemzeit. Die fixe Systemvariable `[TIME]` ist im exakten Format für die Verwendung im KNX System formatiert. Gleiches gilt für die Verwendung der fixen Systemvariable `[DATE]` .

<KNX><GETVALUE=1/2/33></KNX> Wertabfrage der Gruppenadresse `1/2/33`

KNXnet/IP Tunneling Support

Für den Tunneling-Support gelten gleiche Vorgehensweisen, wie zum Routing-Protokoll beschrieben. Es müssen lediglich nachfolgende Schalter in der `CONFIG` -Sektion der `knx.csv` angepasst bzw. verändert werden.

Die Betriebsart `TUNNELING` lässt lediglich nur eine aktive Verbindung zu dem KNXnet/IP-Device zu. Dies kann sich grad bei größeren Installationen nachteilig auswirken. Eine Möglichkeit diesen „Nachteil“ zu umgehen bieten wir Ihnen mit der Betriebsart „`TUNNELING_BRIDGE`“ .

Spezifische Einstellungen für die Betriebsart „TUNNELING“:

CONNECTIONTYPE;TUNNELING

Bei `CONNECTIONTYPE;TUNNELING` wird folgender zusätzlicher Schalter benötigt:

DEVICE_IP;{Gateway IP} legt die IP des KNX-Tunneling-Gerätes fest, mögliche Werte:
`{IP des Gerätes}` oder
„`AUTO`“ für automatisches Suchen (default: `AUTO`)

KNXnet/IP TUNNELING_BRIDGE Support

Der KNXnet/IP-Tunneling_Bridge-Support dient dazu mehrere nomosSysteme über lediglich eine bestehende Tunneling-Verbindung den Zugriff in die KNX-Welt zu ermöglichen. In der Betriebsart „`TUNNELING_BRIDGE`" arbeitet das nomos system quasi als virtueller KNX/IP-Router, der die KNX-Verbindung über eine Tunneling-Verbindung Sicherstellt. Diese Funktionalität kann z.B. auch zur Anlagenkopplung verwendet werden.

Vorgehensweise:

Ausgehend von min. 2 nomos systemen mit aktivierter KNXnet/IP-Einstellung und einem KNXnet/IP-Tunneling-Device müssen folgende Einstellungen vorgenommen werden:

Im Netzwerk muss ein nomos system als „Tunneling-Master" arbeiten. Dieser stellt die Tunneling-Verbindung zu dem KNXnet/IP-Device sicher. Gleichzeitig stellt dieses nomos system eine Multicast-Adresse für den Routing-Support bereit. Über diese Multicast-Adresse können nun weitere Systeme mit dem „Tunneling-Master" kommunizieren. Die weiteren nomos systeme arbeiten hierfür in der Betriebsart „`Routing`".

Bitte stellen Sie sicher, dass die Multicastadresse bzw. der dadurch definierte Multicast-Bereich nur den nomos systemen zur Verfügung steht. Sollte ein weiteres Routing fähiges KNXnet/IP-Device in der Gesamtstruktur erreichbar sein, kann dies zu Störungen bei der Kommunikation führen. Es ist auch darauf zu achten, dass in den Einstellungen keine „`AUTO`" -Funktion aktiviert ist.

Konfiguration „Tunneling Server" (1. nomos system)
`knx.csv` - Folgende Einstellungen definieren das KNXnet/IP-Interface des nomos Systems als Multicast-Gateway mit Tunneling-Zugriff.

```
[CONFIG];
ACTIVE;YES;
DEBUG;YES;
ESF;knx.esf;
PHYSADDR;0.0.254;

**** Einstellung für TUNNELING_BRIDGE ****
```

CONNECTIONTYPE;TUNNELING_BRIDGE Schalter für Betriebsart

DEVICE_IP;AUTO „`AUTO`" sucht das KNXnet/IP-Device, besser ist jedoch die Einstellung einer festen IP-Adresse

MULTICAST_IP;224.0.23.15 Multicast-Adressbereich worüber der Routing-Service bereitgestellt wird.

Konfiguration „Routing" (2. bis n. nomos system)
`knx.csv` - nachfolgende Einstellungen beispielhaft.

```
[CONFIG];
```

```
ACTIVE;YES;
DEBUG;YES;
ESF;knx.esf;
PHYSADDR;0.0.253;

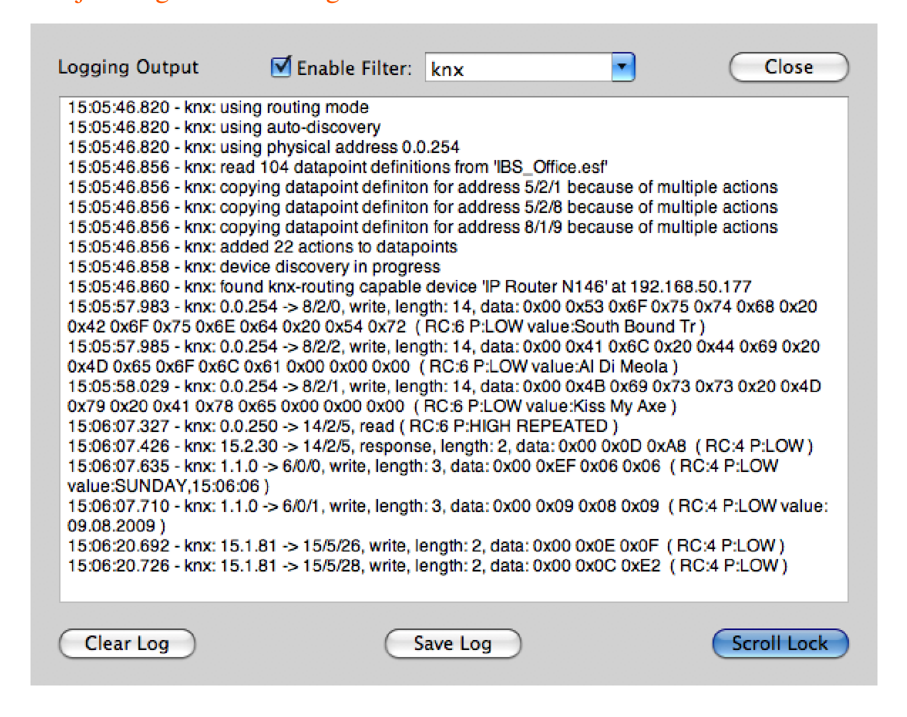
**** Einstellung für ROUTING ****
```

CONNECTIONTYPE;ROUTING Schalter für Betriebsart

MULTICAST_IP;224.0.23.15 Multicast-Adressbereich worüber der Routing-Service bereit gestellt wird.

Alle Geräte im Netz, die das Routing-Protokoll unterstützen, können nun über diese „`MULTICAST_IP`“ `224.0.23.15` eine Gateway-Verbindung zu dem „Tunneling-Master“ aufbauen und gleichzeitig auf den KNX-Bus zugreifen.

Bitte beobachten Sie nach einem Neustart auch die Logausgaben der jeweiligen nomos systeme. Das Log gibt Auskunft über den jeweiligen Verbindungsstatus



4.3 Event Server

Eventserver ermöglichen Protokollsequenzen (Antwortsequenzen) fremder Systeme in die nomos Protokollstruktur zu integrieren bzw. deren Systemzustände zu überwachen. Fremdprotokolle können an dieser Stelle auch direkt verarbeitet werden und bei Bedarf auch direkt Aktionen (nomos Script oder Befehlsketten) auslösen.

Dies wird beispielsweise benötigt, wenn auf ein „Feedback" eines beliebigen Gerätes (z.B. Einschalten des AppleTV über die Apple Remote Fernbedienung, oder der Remote APP) reagiert werden soll. So ist es beispielsweise möglich folgende Funktionalität zu realisieren...

```
... IF APPLE TV = ON, than send SCRIPT TV ON and SWITCH TV
INPUT TO HDMI1
```

Event Definitionen werden im Unterordner `.\{projectPath}\events\` abgelegt. Das Dateiformat ist ebenfalls `*.csv`. Der `[Dateiname].csv` kann frei gewählt werden. Bitte beachten Sie, dass maximal 25 Event-Definitionstabellen (Datein) möglich sind. Ebenso wird für das generelle Verarbeiten der Event- Definitionen eine gesonderte Lizenz benötigt.

Es sind auch mehrere Definitonsdateien pro Port möglich. Bei einem gültigen Event wird die Zeilennummer des Matches und der Dateiname der `.csv` im Logging ausgegeben.

Eventserver öffnen die Verbindung unmittelbar nach dem nomos systemstart und halten diese geöffnet. Eventserver können u.a. auch auf die nomos eigenen Kommunikationsports (Status Broadcast, Log, Kontrollport) angesetzt werden.

```
11:25:23.924 - system: reading file: SystemEvents.csv
11:25:23.926 - system: file SystemEvents.csv has 11 valid lines
```

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\events\
Name:	{EventClass}.csv
Länge:	1000 Zeilen/Einträge
Event:	-
Export:	-
Lizenz:	y

Die `{EventClass}.csv` besteht aus 2 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[TRIGGERS]; Event Definitionen

Aufbau:

Im events-Ordner wird eine `{EventClass}.csv` mit folgendem Inhalt abgelegt:

- **[CONFIG];Sektion**

  ACTIVE;YES Aktiviert/Deaktiviert den Webserver
  „`YES`" (Default)= aktiv,
  „`NO`" = nicht aktiv

  DEBUG;YES Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

TRIGGERIP;{TYP}; Angabe der Quelladresse des entfernten Systems. Dies können folgende Einstellungen sein:

{IP-Adresse} entfernte `IP Adresse` (Auslöser)

INTERNAL für das überwachen der systemeigenen Ports

ANY für Verbindung von jeder beliebigen IP Adresse

TRIGGERPROTOCOL;{TYP}; Angabe des Protokolls, welches empfangen werden soll:

TCP Wartet auf TCP Verbindungen

UDP Wartet auf UDP Verbindungen

TRIGGERPORT;{PORT}; Angabe der Portadresse bzw. des Kommunikationskanals. Die Einstellungen sind abhängig der jeweiligen Einstellungen unter `TRIGGERIP`;

{PORT Adresse} lokaler Port bei Angabe einer IP-Adresse unter `TRIGGERIP`

Bei der Verwendung der Einstellung `TRIGGERIP;INTERNAL` sind folgende Parameter relevant*:

IN Überwacht die eingehenden Pakete des Kommunikationsport (1037)

OUT Überwacht die ausgehenden Pakete des Kommunikationsport (1037)

LOG Überwacht die Ausgaben des LOG Ports (1039)

BROADCAST Überwacht die Ausgaben des Broadcast Status Port (1038)

TRIGGERMODE;{Mode}; Einstellung für den Datentyp des Protokolls.:

ASCII Erwartet die eingenden Daten im ASCII Format

HEX Erwartet die eingehenden Daten im HEX Format.

MATCHING;{Mode}; Einstellung der Abhandlung für das Parsen der `TRIGGERS`. Mögliche Parameter:

NORMAL (Default) - Bei dieser Einstellung erfolgt die Abarbeitung des eingehende Datenpaket sequentiell. Der Eingangsstring wird bei einem erfolgreichen Match entsprechend gekürzt. (subtraktives Verhalten). Einem weiterern Trigger innerhalb der .csv Datei steht für die Matchbedingung nur noch der restliche Eingangsstring zur Verfügung.

FULL Ermöglicht ein zeilenweites Parsen des Eingangsstrings. Der Eingangsstring steht also in voller Länge jedem Triggereintrag zur Verfügung. Es kann auch auf vergleichende Bedingungen getriggert werden.

Beispiel für die Einstellung für den internen Event Support (Verwendung der nomos Standard Ports vorausgesetzt):

TRIGGERIP; INTERNAL und TRIGGERPORT; OUT Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die nomos auf `Port 1037` ausgibt.

TRIGGERIP; INTERNAL und TRIGGERPORT; IN Hat zur Folge, dass die TRIGGERS auf die Befehlssequenzen angewendet werden, die nomos auf `Port 1037` geschrieben werden.

TRIGGERIP; INTERNAL und TRIGGERPORT; LOG Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die nomos auf `Port 1039` ausgibt.

TRIGGERIP; INTERNAL und TRIGGERPORT; BROADCAST Hat zur Folge, dass die TRIGGERS auf die Antwortsequenzen angewendet werden, die nomos auf `Port 1038` bzw. dem Statusport ausgibt. Es werden hierbei nur die Broadcast Ausgaben berücksichtigt, die von dem lokalen nomos Daemon ausgegeben werden.

- **[TRIGGERS];Sektion** Sektion für die TRIGGER Definitionen, die die eingehenden Daten parsen.

  `{Match-String};{Action oder Scriptname};{Optionen};{Bedingung}`

  {Match-String} Die Matchbedingung, auf die die eingehenden Daten geprüft werden. Der Wert, der durch die Parser Optionen (s. Kapitel Parser Optionen (Match Bedingungen) extrahiert wird, wird automatisch als „Argument“ an das aufgerufene Script übergeben. Die Matchbedingung ist **nicht** case sensitiv

  {Action oder Scriptname} nomos Script oder Befehlssequenz, welches bei erfolgreichem Match ausgelöst wird.

  {Optionen} Wird `ONCHANGE` (optional) als Option verwendet, wird eine Aktion nur dann ausgeführt, wenn eine Wertänderung nach dem letzten Trigger erfolgt ist. Die Einstellung ist nur sinnvoll, wenn ein variabler Wert evaluiert werden soll.

  {Match-Bedingung} hier können optional Komperatoren („`>`", "`<`", "`=`") eingesetzt werden. Auch Kombinationen wie zB. „`<>`" (ungleich) oder „`>=`" (kleiner oder gleich) können definiert werden. Werden Match Bedingungen eingesetzt, müssen neben einem erfolgreichem Match auch die Bedingungen erfüllt sein, um eine Aktion auszuführen.

Beispiele f. interne Eventserver:

Bei der Erstellung der Regeln bzw. Trigger für die jeweiligen Eventserver ist es hilfreich, dass LOG zu beobachten. Öffnen Sie die entsprechenden LOG Ausgaben unter Hilfenahme des ScriptingClients oder unter Verwendung von Netcat. Kopieren Sie sich den Teil, den Sie verarbeiten wollen aus dem LOG heraus und bearbeiten Sie den Eintrag entsprechend bzw. fügen Sie die notwendigen Parser Optionen ein. Textstellen, die ignoriert werden sollen, ersetzen Sie durch ein `\%` und Textstellen deren Wert empfangen und verarbeitet werden soll ersetzen Sie durch `\*`. Das Verwenden weiterer Parser Optionen finden Sie in dem entsprechenden Kapitel.

Auswerten der aktuellen Systemlautstärke aus dem Broadcast Status Port

```
<SYS>GETVOL=INT|VOL=81|MUTE=OFF|OK</SYS>
```

TRIGGERIP;INTERNAL und TRIGGERPORT; **BROADCAST**

<SYS>\%|VOL=*|;<SYS><SETVOL=69></SYS>;ONCHANGE;>=70 Matcht auf die Systemlautstärke, die am Statusport ausgegeben wird. Ist der Wert größer oder gleich `70` wird über einen entsprechenden Befehl die Lautstärke auf „`69`" zurückgesetzt. Der Vergleich erfolgt nur bei einer Wertänderung (`ONCHANGE`).

Gem. den Einstellungen sucht der Parser nach einer Zeichenkette, die mit `<SYS>` beginnt und worauf nach einer beliebigen Anzahl von Zeichen eine Zeichenkette `|VOL=` folgt. Nun werden alle folgenden Zeichen bis zum nächsten `|` eingesammelt. Das Match lautet entsprechend „`81`". Der Wert 81 erfüllt die Match-Bedingung `>=70` und somit wird der Befehl `<SYS><SETVOL=69></SYS>` ausgeführt.

Auswerten eines aktuellen Player Status aus dem Broadcast Status Port mit dynamischer Bildung des Scripnamens

```
<ITUNES>GETPLAYERSTATE=PLAYING|
```

TRIGGERIP;INTERNAL und TRIGGERPORT; **BROADCAST**

<ITUNES>GETPLAYERSTATE=*|;<SCRIPT><RUN=SetAMP_\#.myh></SCRIPT>;ONCHANGE;
Matcht auf die Zeichenkette nach `<ITUNES>GETPLAYERSTATE=`. Die Zeichen werden, wie vor erklärt, bis zum `|` gesammelt. Entsprechend der eingehenden Daten lautet der Match „`PLAYING`". Diese Daten werden nun mit dem Parameter `\#` an die Aktion übergeben. In diesem Fall hat dies zur Folge, dass der Skriptname entsprechend ergänzt wird. Aus `SetAMP_\#.myh` wird folglich `SetAMP_PLAYING.myh`. Hierüber können zustandsgesteuert verschiedene Skripte ausgelöst werden.

Zuordnen von Ausgaben der Spracherkennung an entsprechende Funktionen

```
<DEVICE>NAME=iPhone von nomo|USERSAID=decke licht ein|OK</DEVICE>
```

TRIGGERIP;INTERNAL und :txt-font-bold'TRIGGERPORT;' **BROADCAST**

<DEVICE>\%USERSAID=\%licht ein\%|;<KNX><SETVALUE=1/2/1,1></KNX>;
Empfängt Pakete von jedem DEVICE (Name wird ignoriert) und erwartet ein „`licht ein`" als festen Wert. Ist die Bedingung erfüllt, wird die Aktion ausgeführt.

Überwachung einer Eingangsbefehls- Sequenz mit anschließender Aktion

```
<WINDOW><XMIN=340><YMIN=540><XMAX=400><YMAX=120><SELECT=3><text="Welcome to nomos system"><TEXTSIZE=50><CREATE></WINDOW>
```

TRIGGERIP;INTERNAL und TRIGGERPORT; **IN**

<WINDOW>\%<SELECT=3>\%<CREATE>;<WINDOW><SELECT=3><DELAY=3><RELEASE></WINDOW>;
Matcht auf eine eingehende Befehlssequenz, auf die Befehle `<SELECT=3>` und `<CREATE>` der `<WINDOW>` Klasse. Ist diese Bedingung erfüllt, wird das Fenster nach 3 Sekunden geschlossen.

Überwachung einer Ausgabesequenz mit anschließender Aktion

```
<TV>ON=|OK|FRONT=|OK|FSON=|OK|CH=|OK</TV>
```

TRIGGERIP;INTERNAL und TRIGGERPORT; **OUT**

<TV>\%FSON=|OK;<SCRIPT><RUN=MuteAll.myh></SCRIPT>; Matcht auf die Antwort eines ausgeführten `<TV>..<FSON>`-Befehls und führt bei erfolgreichem Match das Script `MuteAll.myh` aus.

Beispiel externer Eventserver:

HTTP Eventserver, der bei Empfang einer entsprechenden URL eine Aktion ausführt.

Nachfolgender Eventserver ermöglicht das übergeben von nomos Protokollsequenzen als URL. Definieren Sie hierfür einen Eventserver mit folgenden Einstellungen:

Bitte beachten Sie, dass der Port 80 für mögliche interne Webserver bereits belegt ist.

[CONFIG];
ACTIVE;YES

TRIGGERIP;ANY;
TRIGGERPORT;2000;
TRIGGERPROTOCOL;TCP;
TRIGGERMODE;ASCII;

[TRIGGERS];
GET /* HTTP;<SYS><\#></SYS>

`TRIGGERIP` kann natürlich auch mit einer fixen IP Adresse belegt werden. In diesem Fall reagiert der Eventserver ausschließlich auf diese Client Adresse. Durch den Parameter `ANY` werden nun Anfragen von jedem möglichen Client bedient.

Öffnen Sie nun einen Browser und senden Sie folgende URL:

http://192.168.1.213:2000/HELLO Die IP Adresse muss natürlich entsprechend Ihrem Zielsystem angepasst werden, oder sie verwenden `localhost (127.0.0.1)` für den lokalen Zugriff.

Folgendes kann nun im Protokoll beobachtet werden:

```
eventmanager: remote event from 127.0.0.1 at port 2000
eventmanager: received data: GET /hello HTTP/1.1\x0D\x0AHost: localhost:2000\x0D
\x0AAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\x0D
\x0AAccept-Language: de-de\x0D\x0AConnection: keep-alive\x0D\x0AAccept-Encoding: gzip,
deflate\x0D\x0AUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/
600.1.17 (KHTML, like Gecko) Version/7.1 Safari/537.85.10\x0D\x0A\x0D\x0A
eventmanager: event valid (line 1 in file Web.csv, arg: hello)
eventmanager: executing: <SYS><\#></SYS> with arg: hello
parser: parsing sequence: <SYS><hello></SYS>
parser: command: SYS - hello
```

Die TRIGGER Regel `GET /\* HTTP` filtert nun den entsprechenden Befehl aus dem URL Header heraus und übergibt der Wert „hello“ an die auszuführende Aktion. Die Kommando- sequenz wird nun zu `<SYS><hello></SYS>` komplettiert und ausgeführt.

4.4 Airport Support (Airfoil)

Das nomos system unterstützt die Steuerung externer Lautsprecher wie zB über den Apple Airport Express. Die Nutzung dieser Funktion setzt den Einsatz der Airfoil-Software (Mac only) voraus. Die Software kann unter folgender URL bezogen werden: http://rogueamoeba.com/airfoil/mac/.

Die Installationshinweise finden Sie ebenfalls auf der Homepage des Herstellers. Bitte beachten Sie, dass die Anwendungen, deren Ton übertragen werden soll, entsprechend in den Airfoil-Einstellungen angemeldet werden müssen (erscheinen im PullDown-Menü von Airfoil).

Der Airport Kommandosatz befindet sich in der `<SYS>` Klasse mit folgenden zusätzlichen Befehlen:

Kommando	Argument	Beschreibung
GETSPEAKERLIST		listet alle verfügbaren Airports mit Namen auf ("Computer" ist der lokale Ton des Apple Computers)
GETSPEAKERSOURCE		gibt die aktuelle Applikation bzw. den aktiven Eingang aus, deren Ton gerade gestreamt wird
SETSPEAKERSOURCE#	{Appname}	wählt die Applikation aus, deren Ton gestreamt werden soll
SETSPEAKERSOURCEDEVICE#	{Eingang}	wählt den Audio-Eingang aus, deren Ton gestreamt ICE# werden soll
GETAUDIODEVICELIST		listet alle verfügbaren Audio-Eingänge auf
GETSPEAKERVOL#	{Airport-Name}	gibt die aktuelle Lautstärke des gewählten Airports aus
SETSPEAKERVOL#	{Airport-Name,} {0...100}	setzt die Lautstärke des gewählten Airports absolut, wird der Name weggelassen, wird die Lautstärke für alle Airports (inkl. "Computer") gesetzt
SPEAKERVOLUP#	{Airport-Name,} {0...100}	erhöht die Lautstärke um den angegebenen Wert, der Name ist auch hier optional
SPEAKERVOLDN#	{Airport-Name,} {0...100}	senkt die Lautstärke, sonst wie `SPEAKERVOLUP`
ENABLESPEAKER#	{Airport-Name}	schaltet den gewählten Airport ein
DISABLESPEAKER#	{Airport-Name}	schaltet den gewählten Airport aus
ENABLEALLSPEAKERS		schaltet alle Airports ein
DISABLEALLSPEAKERS		schaltet alle Airports aus
GETSPEAKERSTATE#	{Airport-Name}	liefert den Status des gewählten Airports: `ENABLED` / `DISABLED` / `NOT_AVAILABLE`

Wir weisen ausdrücklich darauf hin, dass mit dieser Lösung keine Multiroom-Anwendungen zu realisieren sind. Die Airport-Steuerung dient lediglich der einfachen Einbindung von max. 3-4 zusätzlichen Audiozonen. Zu beachten ist auch, dass der empfangene Stream etwas zeitversetzt gesendet wird. Es ist z.B. nicht möglich, den Audiostream zu synchronisieren. Die Audio Wiedergabe von z.B. TV-Sendungen oder anderen Anwendungen mit Audio/Video-Übertragungen ist durch diesen Zeitversatz nicht Lippensynchron und daher nicht zu empfehlen.

4.5 eyeTV - Streaming Support

Sofern das System über einen TV Tuner verfügt, besteht die Möglichkeit den unter eyeTV abgespielten TV- Kanal in das Netzwerk zu streamen (Mac only). Der Stream kann z.B. über die kostenlose Software VLC empfangen werden. Das notwendige eyeTV-Plugin wird automatisch mit der nomos Installationsroutine installiert.

Die Steuerung des Streaming Servers erfolgt in der Klasse STREAMER mit folgenden Befehlen:

Kommando	Argument	Beschreibung
ON		startet die Streaming-Engine und ggf. EyeTV (minimiert)
OFF		beendet die Streaming-Engine, EyeTV läuft weiter
CH#	{Kanalnummer} oder {Kanalname}	wählt die Streaming-Quelle, dieser Befehle funktioniert analog zum `SELECT` -Befehl in der Windows-Klasse. Alle nachfolgenden, kanalbezogenen Befehle beziehen sich auf den zuletzt mit `CH` ausgewählten Kanal
GETCHLIST		liefert eine Liste mit allen zur Zeit verfügbaren Kanälen und der entsprechenden Kanalnummer.
ADDTARGET#	{IP:Port}	weist der zuvor mit `CH` gewählten Quelle ein Ziel zu und beginnt sofort mit dem UDP Streaming zur angegebenen Adresse. Es wird eine eindeutige Streaming-ID zurückgegeben. Auf dem Client dem VLC bitte als URL `udp://@:{Portnummer}` übergeben.
REMOVETARGET#	{IP:Port} oder {Streaming-ID} oder {IP}	stoppt das Streaming des per `IP:Port` oder `ID` angegeben Streams. Wird nur die `IP` angegeben, werden alle an diese `IP` gerichteten Streams gestoppt.
REMOVEALLTARGETS		stoppt alle Streams
GETTARGETSFORIP#	{IP}	listet alle an die `IP` gerichteten Streams auf
GETTARGETSFORCH#	{Kanalnummer} oder {Kanalname}	listet alle `IP-Adressen` auf, an die der Kanal gestreamt wird
GETALLTARGETS		listet alles auf, was gestreamt wird

Grundsätzlich:

Die Kanalnummern der jeweiligen Sender sind nicht festgelegt, sondern können sich beim Neustart von EyeTV oder nomos ändern. Deshalb ist es sinnvoller, den Sender anhand des Namens auszuwählen.Es stehen immer alle Sender eines Transponders zur Verfügung. Mit 4 Tunern sind so also 16 Programme gleichzeitig möglich, wenn man die Sender in EyeTV geschickt wählt. Ein USB-Stick im Dual-Mode liefert also 8 Programme gleichzeitig. Sollte es bei mehr als 2 TV-Bildern in EyeTV zu Streaming Aussetzern kommen, liegt das an der hohen Prozessorlast, die EyeTV erzeugt. Wir empfehlen dann einfach die nicht lokal notwendigen TV-BIlder ins Dock minimieren.

Streaming an Broadcast-Adressen ist möglich. Die Zahl der Streams ist nur durch die System Hardware bzw. das Netzwerk beschränkt.‘

Zusätzlich ist TCP-Streaming ohne Steuerungsbefehle möglich: Die Portnummer entspricht dabei der Kanalnummer+61000. Wenn z.B. RTL auf Kanal 5 läuft, kann der VLC sich dieses Programm mit der URL `tcp://{IP-Adresse}:61005` "abholen". Die Zahl der TCP-Streams ist ebenfalls nicht begrenzt. Sollte es beim TCP-Streaming zu kurzen (Ton-)Aussetzern kommen, bitte in den Einstellungen des VLC das Caching erhöhen, z.B. auf 500ms.

Beispiele:

<STREAMER><GETCHLIST></STREAMER> Ausgeben aller z.Z. verfügbaren DVB-Streams:

Antwort: <STREAMER>GETCHLIST=CH1=Das Erste | CH2=Bayerisches FS Süd | CH3=hr-fernsehen | CH4=Bayerisches FS Nord | CH5=WDR Köln | CH6=BR-alpha* | CH7=SWR | OK</STREAMER>

<STREAMER><CH=Das Erste><ADDTARGET=192.168.1.240:45678></STREAMER> UDP-Streaming des Programms „Das Erste“ an IP 192.168.1.240, Port 45678:

Antwort: <STREAMER>CH=1 | Das Erste | OK | ADDTARGET=567739290 | OK</STREAMER>

4.6 Apple Remote Support

iTunes- und AppleTV-Steuerung. Es wird für jede in der `remote.csv` definierte dynamische Klasse beim Systemstart versucht, eine Verbindung herzustellen. Sofern innerhalb der iTunes Anwendung oder aber beim AppleTV die Privatfreigabe/Homesharing aktiviert wurde, erkennt das nomos system das völlig automatisch. Wenn dies erfolgt ist, steht der rudimentäre Remote-Befehlssatz (abgesehen von den Button-Befehlen, die immer nur beim AppleTV funktionieren) zur Verfügung.

Wurde das Pairing durchgeführt (s. weiteren Textverlauf), steht ein erweiterter Befehlssatz für die Steuerung zur Verfügung. Der erweiterte Befehlssatz ist jedoch ausschließlich für die Steuerung eines „entfernten" iTunes verfügbar. Einem AppleTV, egal ob via Privatfreigabe/Homesharing verbunden, oder aber über die Pairing Methode, steht immer nur der rudimentäre Remote-Befehlssatz zur Verfügung.

Da das nomos system anhand des Protokollverhaltens nicht erkennen kann, um welches System es sich bei dem Zielsystem handelt (ATV oder ein nicht gepairtes iTunes), werden die für das Homesharing Protokoll ungültigen Befehle zurückgewiesen bzw. eine Fehlermeldung generiert.

Für die Steuerung eines „entfernten" iTunes ist die Installation der nomos system Software auf dem Zielsystem nicht erforderlich.

Das LOG gibt Auskunft, welches Protokoll für das zu steuernde Gerät aktiviert wurde:

Eine Verbindung zu dem Gerät konnte nicht aufgebaut werden

```
remote: ERROR: connection to device MBA (IP: 192.168.1.20) failed
```

Auszug aus dem LOG für ein verbundenes AppleTV (definiert als ATV) und aktivierter Privatfreigabe/ Homesharing

```
remote: successfully connected to device ATV (IP: 192.168.1.21), using home sharing protocol with
reduced command set
```

Auszug aus dem LOG für ein verbundenes iTunes (definiert als MBP) und aktivierter Privatfreigabe/ Homesharing

```
remote: successfully connected to device MBA (IP: 192.168.1.20), using home sharing protocol with
reduced command set
```

Auszug aus dem LOG für ein verbundenes iTunes (definiert als MBP) nach erfolgtem Pairing. Hier steht nun der vollständige Befehlssatz zur Verfügung.

```
remote: successfully connected to device MBA (IP: 192.168.1.20)
```

Das erweiterte Protokoll verfügt ebenfalls über einen Befehlssatz für das Steuern „entfernter Lautsprecher" (Airstream), sofern diese dem entfernt zu steuernden iTunes bekannt sind. Die Airstream Steuerung ist hierbei nicht zu verwechseln mit der Airfoil Steuerung.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	remote.csv
Länge:	25 Zeilen/Einträge
Event:	BC, Status
Export:	nein
Lizenz:	ja

Die `remote.csv` **besteht aus 2 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DEVICES]; Liste der fernzusteuernden Geräte. Es können maximal 25 Geräte angemeldet werden

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES; Schaltet das Remote Modul
  „`YES`" = ein (default),
  „`NO`" = aus

  DEBUG;YES; Schalter für
  „`YES`" = erweiterte Logausgaben (Busmonitor)
  „`NO`" = standard Logausgaben (default)

- **[DEVICES];Sektion**

  {Gerätename};{IP-Adresse}; Zuordnung der Geräte IP Adresse zu der entsprechenden dynamischen Klasse für den Protokoll Support.

Erläuterungen:

{Gerätename} Name des zu steuernden Gerätes. Der hier vergebene Name beschreibt die Klasse, mit der das Gerät angesprochen werden kann.

{IP-Adresse} IP Adresse des zu steuernden AppleTV oder Apple zw. Windows Computer mit installierter iTunes Software.

Beispiele für die Definitionen [DEVICE] Sektion:

ATV;192.168.1.21 Definiert ein Device an IP-Adresse 192.168.1.21 mit den Namen ATV. Das Gerät ist nun über die Klasse <ATV> zu steuern.

MBP;192.168.1.20 Definiert ein Device an IP-Adresse 192.168.1.20 mit den Namen MBP. Das Gerät ist nun über die Klasse <MBP> zu steuern.

Es ist an dieser Stelle völlig egal, ob es sich bei dem zu steuernden Gerät um ein AppleTV oder um ein „entferntes" iTunes System handelt. Das nomos system erkennt das Gerät und das mögliche Protokoll automatisch.

Die unter `{Gerätename}` definierte Klasse darf nur einmalig im System vorhanden sein. Der Name darf auch nicht einem bereits reservierten Klassennamen entsprechen. Das zu steuernde Gerät muss entweder die Privatfreigabe/Homesharing aktiviert haben und/oder gepairt werden.

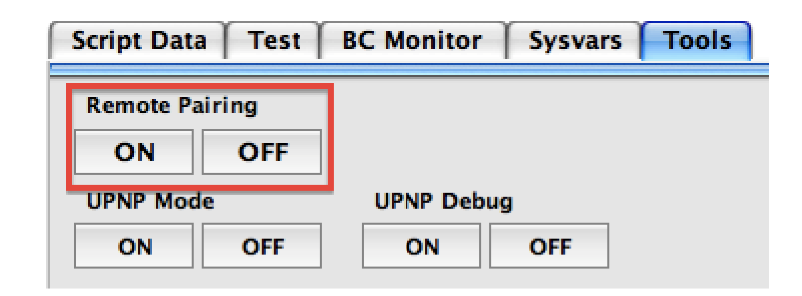
Das Pairing:

Nachdem Sie die Geräte bzw. das Zielsystem der „entfernten" iTunes Installation in der `remote.csv` registriert haben, können Sie das Pairing einleiten. Die iTunes Anwendung muss für das Pairing auf dem Zielsystem geöffnet sein. Das Pairing muss grundsätzlich nur einmalig erfolgen.

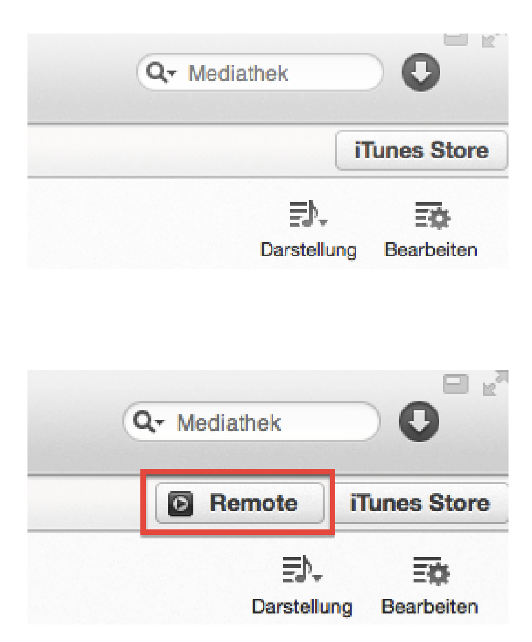
Um das Pairing einzuleiten, steht folgender Befehl zur Verfügung:

<SYS><REMOTEPAIRING=ON></SYS> Startet das Remote Pairing

Ebenso ist im ScriptingClient eine entsprechende Funktion zum Einleiten des Remote Pairing verfügbar



Ist der Remote Pairing Mode aktiv, wird das nomos system in iTunes bzw. im AppleTV sichtbar:



Bei einem AppleTV finden Sie diese Option in den `Einstellungen` unter `Fernbedienung` und dort entsprechend unterhalb `iOS Fernbedienungen`.

Nach der Auswahl werden Sie aufgefordert einen 4-stelligen Code einzugeben. Dieser Code kann frei gewählt werden und hat keinerlei weitere Auswirkungen auf die Pairing Prozedur. Ab diesem Zeitpunkt stehen die definierten Klassen mit ihren Befehlen zur Verfügung. Um den Verbindungs-aufbau kümmert sich das nomos system im Hintergrund (siehe Log). Im LOG erscheinen eispielhaft entsprechende Meldungen:

Meldung, dass das Remote Pairing aktv ist

```
remote: pairing enabled
```

Meldung, dass das Pairing erfolgreich ausgeführt wurde.

```
remote: device "iTunes/11.1.3" (IP: 192.168.1.20) successfully paired
```

Bitte vergessen Sie nicht, den Pairing Mode wieder zu deaktivieren. Dies erfolgt entweder über den entsprechenden Button in dem ScriptingClient, oder aber über den direkten Befehl:

<SYS><REMOTEPAIRING=OFF></SYS> Beendet das Remote Pairing

Nach dem Neustart des Daemons ist der Pairing Mode ebenfalls deaktiviert.

Nach erfolgreicher Registrierung können die Geräte nun gesteuert werden. Die Statusaktualisierung der Remote-Geräte erfolgt automatisch im Broadcast, wenn sich der Play-Status und/oder der aktuelle Titel des entsprechenden Gerätes ändert.

```
BC:<MBA>GETINFO=PLAYING|SHUFFLE=OFF|REPEAT=OFF|TITLE=The A Team|ARTIST=Ed Sheeran|ALBUM=The A Team -
EP|GENRE=Singer/Songwriter|OK</MBA>
```

Ebenso ist das aktuelle Cover über den Mini- Webserver abrufbar.

http://{IP}:{Portnummer}/{Class}_current.jpg Zeigt das Cover des laufenden Titels der verwendeten Klasse, wie in der `remote.csv`

Ein AppleTV lässt sich bedingt durch das eingeschränkte Protokoll nur über das OSD sinnvoll steuern. Eine protokollgestützte Titel/Album oder Playlisten Anwahl ist leider nicht möglich.

Es existiert eine dynamische Befehlsklasse [Remote] mit folgenden Befehlen:

Kommandos, die ein Argument erwarten, können auch Listen verarbeiten.

Kommando	Argument	Beschreibung
GETINFO		liefert Informationen über den aktuellen Player Status
NEXT		springt zum nächsten Titel
PREV		springt zum vorigen Titel
PAUSE		pausiert den aktuellen Titel
PLAY		spielt den aktuellen Titel ab
TOGGLEPLAY		toggelt zwischen PLAY und PAUSE
STOP		stoppt den aktuellen Titel
SETSHUFFLE#	{ON/OFF}	schaltet den Modus Shuffle der aktuellen Playlist ein o. aus
SETREPEAT#	{ONE/ALL/OFF}	setzt den Repeatmodus der aktuellen Playlist
SETPTIME#	{1...x}	springt zur angegebenen Spielzeit
GETCTIME		liefert die gesamte Spielzeit in Sekunden
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
BUTTON_UP		simuliert einen Tastendruck der AppleRemote
BUTTON_DOWN		simuliert einen Tastendruck der AppleRemote
BUTTON_LEFT		simuliert einen Tastendruck der AppleRemote
BUTTON_RIGHT		simuliert einen Tastendruck der AppleRemote
BUTTON_SELECT		simuliert einen Tastendruck der AppleRemote
BUTTON_SELECTLONG		simuliert einen Tastendruck der AppleRemote (Kontext Menu)
BUTTON_MENU		simuliert einen Tastendruck der AppleRemote
- extended commands -		Erweiterter Befehlssatz, nicht für Verbindungen über Home Sharing
ADDIDTOQUEUE#	{ID}	hängt die ID x an die Up-Next-Queue an
ADDTITLETOQUEUE#	{Name}	hängt den Titel x an die Up-Next-Queue an
ADDARTISTTOQUEUE#	{Name}	hängt den Interpreten x an die Up-Next-Queue an
ADDALBUMTOQUEUE#	{Name}	hängt das Album x an die Up-Next-Queue an

Kommando	Argument	Beschreibung
CLEARQUEUE		löscht die Up-Next Queue
DISABLEALLSPEAKERS		schaltet alle Audio-Ausgabegeräte aus
DISABLESPEAKER#	{Speakername}	schaltet `{Speakername}` aus
ENABLEALLSPEAKERS		schaltet alle Audio-Ausgabegeräte ein
ENABLESPEAKER#	{Speakername}	schaltet `{Speakername}` ein
GETALLALBUMS		liefert eine Liste aller Alben
GETALLARTISTS		liefert eine Liste aller Interpreten
GETALLIDSOFPLAYLIST#	{Playlist}	liefert alle IDs in `{Playliste}` (max. 250)
GETALLTITLESOFPLAY LIST#	{Playlist}	liefert alle Titel in `{Playliste}` (max. 250)
GETALLARTISTSOFPLAY LIST#	{Playlist}	liefert alle Interpreten in `{Playliste}` (max. 250)
GETALLALBUMSOFPLAY LIST#	{Playlist}	liefert alle Alben in `{Playliste}` (max. 250)
GETALLPLAYLISTS		liefert eine Liste aller vorhanden Playlisten
GETIDALBUM#	{ID}	liefert das Album zur angegebenen `{ID}`
GETIDARTIST#	{ID}	liefert den Interpreten zur angegebenen `{ID}`
GETIDTITLE#	{ID}	liefert den Titel zur angegebenen `{ID}`
GETSPEAKERLIST		liefert eine Liste aller erkannten Audio-Ausgabegeräte
GETSPEAKERSTATE#	{Speakername}	liefert den Status (ON bzw. OFF) von `{Speakername}`
GETSPEAKERVOL#	{Name}	liefert die Lautstärke eines Airtunes Lautsprechers relativ zur Master Lautstärke
GETVOL		liefert die aktuelle Applikationslautstaerke
PLAYID#	{ID}	spielt den Titel `{ID}` ab
PLAYPLAYLIST#	{Name}	spielt die Playliste `{Name}` ab
SEARCHALBUM#	{Album}	liefert alle IDs, die zu `{Album}` gehoeren
SEARCHARTIST#	{Suchstring}	liefert eine Liste der Interpreten zurück, auf die der Suchstring passt (Limit: 50)
SEARCHTITLE#	{Suchstring}	liefert eine Track-ID -Liste der Titel zurück, auf die der Suchstring passt (Limit: 50)
SETSPEAKERVOL#	{Name}, {Wert 0-100}	setzt die Lautstärke eines Airtunes Lautsprechers relativ zur Master Lautstärke
SETVOL#	{0...100}	setzt die Lautstärke auf 0-100%

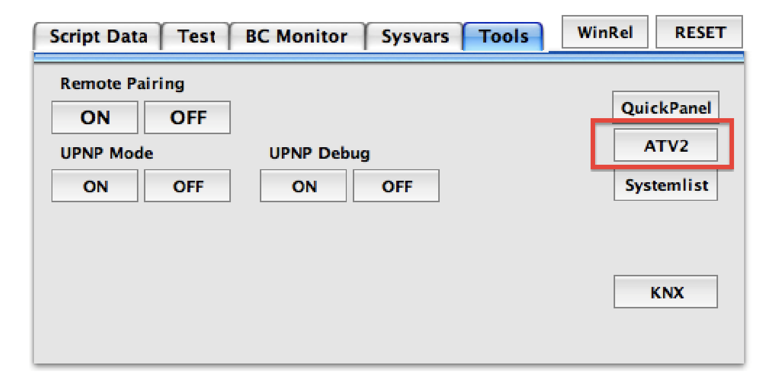
Dazu gibt es die passenden SYS-Befehle:

Kommando	Argument	Beschreibung
GETREMOTEDEVICELIST		liefert eine Liste aller in der `remote.csv` definierten Geräte
REMOTEPAIRING#	{ON}/{OFF}	Startet {ON} /Beendet {OFF} den Pairing Mode für iTunes und AppleTV in der remote Klasse

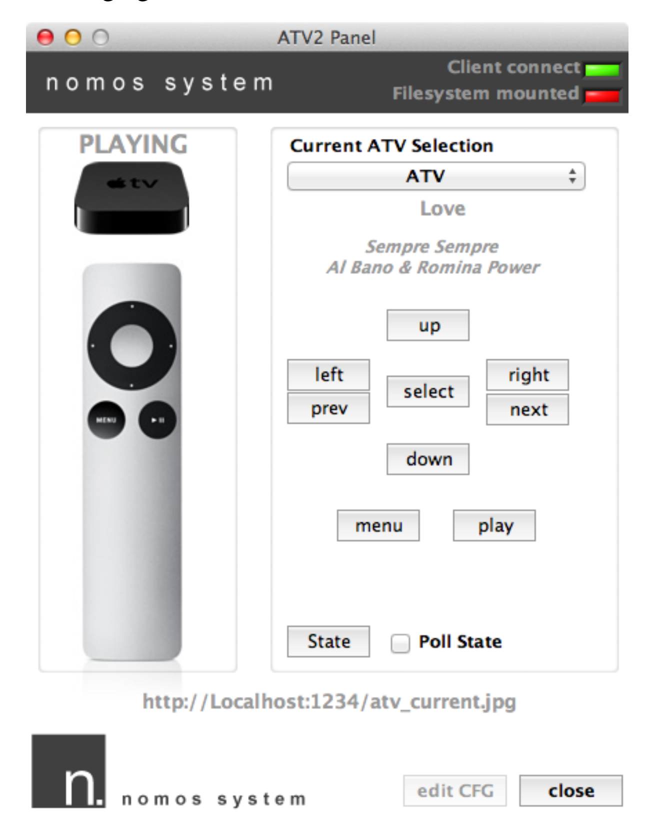
Beispiele AppleTV:

In dem folgendem Beispiel wurde ein AppleTV mit der Klasse `<ATV>` registriert. Sollte sich ein AppleTV im Standby Modus befinden, wird das AppleTV beim Empfang eines Befehls automatisch geweckt. Bitte beachten, dass das AppleTV nur mit dem reduzierten Homesharing Befehlssatz zu steuern ist.

Im ScriptingClient ist ebenfalls eine Steuerkonsole vorhanden. Hierüber können Sie die Möglichkeiten der Steuerung auf einfache Weise testen.



Nach der Auswahl erscheint ein Dialogfenster (Anzeige kann abweichen) dieses ermöglicht die Steuerung aller registrierten Geräte über das Homesharing Protokoll. Ebenso wird der aktuelle Zustand und die Art Informationen zur Anzeige gebracht.



<ATV><BUTTON_SELECT></ATV> Simuliert den Tastendruck der OK Taste der Remote Fernbedienung.

<ATV><BUTTON_MENU></ATV> Simuliert den Tastendruck der Menu Taste der Remote Fernbedienung.

<ATV><BUTTON_LEFT></ATV> Simuliert den Tastendruck der „links“ Taste der Remote Fernbedienung.

<ATV><GETINFO></ATV> Abfrage des aktuellen Gerätestatus. Wird ebenfalls per Event autom. generiert.

Antwort: <ATV>GETINFO=PAUSED | SHUFFLE=OFF | REPEAT=OFF | TITLE=C’est Facile | ARTIST=Al Bano & Romina Power | ALBUM=Sempre Sempre | GENRE= | OK</ATV>

Beispiele iTunes:

In dem folgendem Beispiel wurde ein „entferntes“ iTunes mit der Klasse `<MBA>` registriert.

<MBA><GETALLPLAYLISTS{%I2,5}></MBA> Leitet die Abfrage der Ausgabe einer Playliste ab Index 2 mit einer Länge von 5 Elementen ein.

Antwort: <MBA>GETALLPLAYLISTS{%I2,5}=Musik | Leihobjekte | Filme | TV-Sendungen | Podcasts | OK</MBA>

<MBA><GETALLPLAYLISTS{%I2,5}></MBA> Leitet die Abfrage der Ausgabe einer Playliste ab Index 2 mit einer Länge von 5 Elementen ein.

<MBA><PLAYPLAYLIST=Musik></MBA> Startet das Abspielen der Playliste Musik

Beispiele iTunes Lautsprecher:

In dem folgendem Beispiel wurde ein „entferntes“ iTunes mit der Klasse `<MBA>` und ein AppleTV mit der Klasse `<ATV>` registriert.



<MBA><GETSPEAKERLIST></MBA> Leitet die Abfrage der möglichen „entfernten“ Lautsprecher ein, mit der beispielhaften Antwort:

<MBA>GETSPEAKERLIST=Computer|Apple TV|OK</MBA>

<MBA><ENABLEALLSPEAKERS></MBA> Aktiviert alle registrierten „entfernten“ Lautsprecher inkl. der Ausgabe am Computer

<MBA><SETSPEAKERVOL=Apple TV,80></MBA> Setzt die Lautstärke des Apple TV auf 80%, die Lautstärke bezieht sich dabei immer auf den Maximalwert, der mit SETVOL vorgegeben wurde (Master-Lautstärke).

Systembedingt muß immer ein Ausgabegerät aktiv sein. Werden alle Lautsprecher abgeschaltet, aktiviert iTunes automatisch die lokale Audio-Ausgabe.

4.7 Sonos Streaming Player Support

Steuerung von Sonos Musik Playern. Es wird für jede, in der `sonos.csv` definierte Klasse beim Systemstart versucht, eine Verbindung herzustellen. Die Konfiguration erfolgt ähnlich der `remote.csv`, wie vor beschrieben. Ein Pairing o.ä. ist hier nicht erforderlich.

Da die Sonos-Engine auf UPnP aufbaut, ist zur Steuerung eine UPnP-Lizenz erforderlich.

Sonos arbeitet (bis auf Radio-Sender, die mit einem `PLAYFAVORITE={Sendername}` aufgerufen werden können) mit einer Queue, die zunächst gefüllt werden muss. Die `PLAY*` -Befehle machen dies automatisch.

Die Beschränkung der `GETALL*`-Befehle auf 100 Einträge in der Ausgabe, lässt sich mit dem Listen Formatierer `{%lx,y}` umgehen, es werden jedoch in keinem Fall mehr als 100 Einträge ausgegeben. Pay - Services (Spotify/Netstreams) werden ebenso unterstützt, wie die Status Broadcast Ausgabe und der Coversupport.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	sonos.csv
Länge:	25 Zeilen/Einträge
Event:	BC, Status
Export:	-
Lizenz:	ja

Die `remote.csv` besteht aus 2 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DEVICES]; Liste der fernzusteuernden Geräte. Es können maximal 25 Geräte angemeldet werden

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES; Schaltet das Sonos Modul
  „`YES`" = ein (default),
  „`NO`" = aus

  DEBUG;YES; Schalter für
  „`YES`" = erweiterte Logausgaben (Busmonitor)
  „`NO`" = standard Logausgaben (default)

- **[DEVICES];Sektion**

  {Gerätename};{IP-Adresse}; Zuordnung der Geräte IP Adresse zu der entsprechenden dynamischen Klasse für den Protokoll Support.

  Erläuterungen:

  {Gerätename} Name des zu steuernden Gerätes. Der hier vergebene Name beschreibt die Klasse `{CLASS}` , mit der das Gerät angesprochen werden kann. Wenn als Klassenname der Zonenname des Players eingetragen wird, muss `{IP-Adresse/Grupenname}` nicht angegeben werden.

{IP-Adresse/Grupenname} IP Adresse oder Gruppenname des zu steuernden Sonos Gerätes

Die BC Status Ausgabe

Alle Statusmeldungen werden ohne gezielte Abfrage permanent im Status Broadcast gesendet. Die Daten können mittels EventServer ausgewertet und verarbeitet werden.

Ausgabe eines Player Status bei einem Internet Radio Stream

```
<SON_POOL>STATE=PLAYING|REPEAT=OFF|SHUFFLE=OFF|CROSSFADE=OFF|TRACKS=4|INDEX=1|SOURCE=STREAM|
ARTIST=ANTENNE  BAYERN  -  Verkehrsservice|STREAM=ANTENNE  BAYERN|TITLE=ANTENNE  BAYERN|ALBUM=|OK</
SON_POOL>
```

Ausgabe eines Player Status bei einer File Wiedergabe

```
<SON_POOL>STATE=PLAYING|REPEAT=OFF|SHUFFLE=OFF|CROSSFADE=OFF|TRACKS=1|INDEX=1|SOURCE=FILE|
DURATION=0:14:17|TITLE=Do You Feel Like We Do|ARTIST=Peter Frampton|ALBUM=Frampton Comes Alive|OK</
SON_POOL>
```

Ausgabe der Parameter für die Lautstärke- und Klangeinstellungen

```
<SON_POOL>GETVOL=44|MUTE=OFF|BASS=0|TREBLE=0|LOUDNESS=ON|OK</SON_POOL>
```

Auch das aktuelle Cover des aktuellen Titels, sofern vorhanden, wird über den Mini Webserver bereitgestellt und kann unter folgender URL abgerufen werden:

http://{SYSTEM IP}:{WEBSERVER PORT}/{SONOS_CLASS}_current.jpg

Es existiert eine Befehlsklasse [SONOS] mit folgenden Befehlen:

Kommando	Argument	Beschreibung
PLAY		spielt den aktuellen Titel ab
PAUSE		pausiert den aktuellen Titel
STOP		stoppt den aktuellen Titel
NEXT		springt zum nächsten Titel
PREV		springt zum vorherigen Titel
JUMPTOINDEX#	{1...x}	springt zum x-ten Titel in der Queue
GETVOL		liefert die aktuelle Applikationslautstärke
SETVOL#	{0...100}	Setzt die Lautstärke auf 0-100%
SETBALANCE#	{-100...0...100}	Setzt die Balance (-100)Links - (+100)Rechts
GETBALANCE		Liefert die aktuelle Balance
MUTEON		Stummschaltung EIN
MUTEOFF		Stummschaltung AUS
LOUDNESSON		Loudness EIN
LOUDNESSOFF		Loudness AUS
SETTREBLE#	{-10...0...10}	EQ Setzt die Höhen
SETBASS#	{-10...0...10}	EQ Setzt die Bässe
SETPTIME#	{Sekunden}	springt zur angegebenen Spielzeit
GETPTIME		liefert die aktuelle Spielzeit in Sekunden
GETCTIME		liefert die gesamte Spielzeit in Sekunden
SETREPEAT#	{ON/OFF,1/0,YES/NO}	schaltet den Wiederholen Modus der aktuellen Playlist ein o. aus
SETSHUFFLE#	{ON/OFF,1/0,YES/NO}	schaltet den Zufall Modus der aktuellen Playlist ein o. aus
SETCROSSFADE#	{ON/OFF,1/0,YES/NO}	schaltet den Überblenden Modus der aktuellen Playlist ein o. aus

Kommando	Argument	Beschreibung
GETALLPLAYLISTS		liefert eine Liste aller vorhanden Playlisten
PLAYPLAYLIST#	{Name}	Löscht die Queue und spielt Playliste {Name} sofort ab
ENQUEUEPLAYLIST#	{Name}	fügt Album {Name} ans Ende der Queue an
GETALLTITLESOFPLAY LIST#	{Name}	gibt alle Titel der Playliste {Name} aus
GETALLARTISTSOFPLAY LIST#	{Name}	gibt alle Interpreten der Playliste {Name} aus
GETALLALBUMSOFPLAY LIST#	{Name}	gibt alleAlben der Playliste {Name} aus
GETALLALBUMS		liefert eine Liste aller Alben
PLAYALBUM#	{Name}	Löscht die Queue und spielt Album {Name} sofort ab
ENQUEUEALBUM#	{Name}	fügt Album {Name} ans Ende der Queue an
GETALLTITLESOF ALBUM#	{Name}	gibt alle Titel des Albums {Name} aus
GETALLARTISTSOF AL-BUM#	{Name}	gibt alle Interpreten des Albums {Name} aus
GETALLFAVORITES		gibt alle Favoriten aus
PLAYFAVORITE#	{Name}	spielt Favorit {Name} sofort ab
GETALLTITLESOFQUEUE		gibt alle Titel der Queue aus
GETALLARTISTSOFQUEUE		gibt alle Interpreten der Queue aus
GETALLALBUMSOFQUEUE		gibt alle Alben der Queue aus
CLEARQUEUE		löscht die Queue
LEDON		schaltet die LED des Sonos Gerätes ein
LEDOFF		schaltet die LED des Sonos Gerätes aus
ADDMEMBER#	{Name}	fuegt ein weiteres Sonos Gerät mit dem Klassennamen {Name} als Wiedergabegeraet hinzu (Quellen-Synchronisierung),"x" muss in der sonos.csv definiert sein.
REMOVEMEMBER#	{Name}	entfernt das Sonos Geraet mit dem Klassennamen
STARTLINEINSTREAM#	{Name}	streamt das Signal vom Line-In Eingang an das Gerät mit dem Klassennamen "x", "x" muss in der sonos.csv definiert sein
STOPLINEINSTREAM#	{Name}	stoppt den Stream an das Gerät "x"
PLAYLINEIN		spielt das Signal vom Line-In Eingang lokal ab
SETLINEINLEVEL#	{-15 - 15}	-15 - (+)15 - passt die Empfindlichkeit des Line-In Eingangs an die analoge Quelle an
GETLINEINLEVEL		liefert die eingestellte Empfindlichkeit des Line-In Eingangs

Beispiele:

<{SONOS_CLASS}><GETALLPLAYLISTS></{SONOS_CLASS}> liefert eine Liste alle vorhandenen Playlisten

<{SONOS_CLASS}><PLAYPLAYLIST=Sonos2></{SONOS_CLASS}> Startet das Abspielen der Playlist mit dem Namen Sonos2

<{SONOS_CLASS} ><SETVOL=23></{SONOS_CLASS}> Setzt die Lautstärke des Sonos Device auf 23%

<SON_POOL><JUMPTOINDEX=7></SON_POOL> Springt zum siebten Titel der aktuellen Palyliste oder dem aktuellen Album.

4.8 Z-Wave Support

Z-Wave ist ein drahtloser Kommunikations-Standard, der von der Firma Sigma Designs und der Z-Wave Alliance für die Heimautomatisierung entwickelt wurde. Die hardwarenahen Protokollschichten sind seit 2012 von der ITU-T als Standard G.9959 definiert [1]

Spezifikationen:

- Übertragungsrate: 9.600 bit/s oder 40 Kbit/s
- Modulation: 2-FSK
- Reichweite: Bis zu 200 Meter im Freifeld, in Gebäuden, abhängig von den verwendeten Baumaterialien, deutlich weniger; typisch um 30 m
- Frequenzband: Z-Wave nutzt das ISM-Band im Bereich um 900 MHz, in den USA sind es 908,42 MHz, in Europa 868,42 MHz.

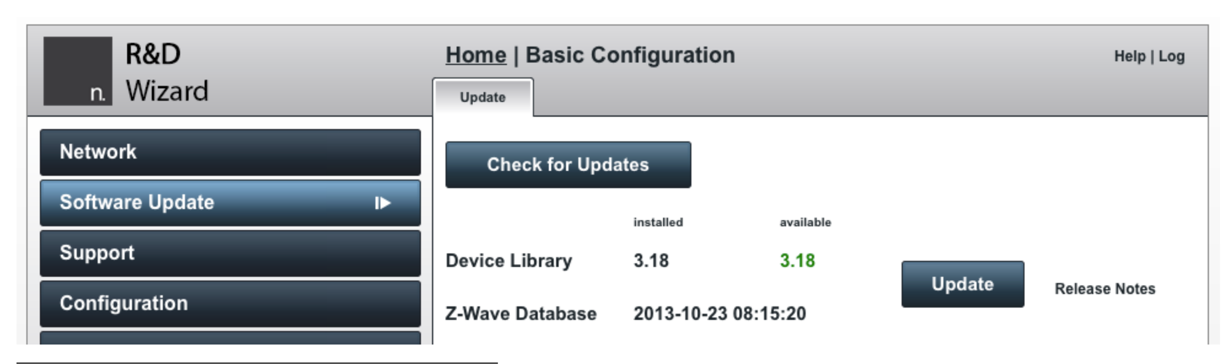
Z-Wave ist z.Z nur für die nomos Box und dem nomos Pi verfügbar. Für die Anbindung wird eine entsprechende Erweiterung (USB Stick) benötigt. Wir empfehlen hierfür den **Aeon Labs** USB Adapter. Sie können diesen Adapter und weitere Komponenten im online Store bei http://www.zwaveeurope.com beziehen. Die Daten des installierten Controllers werden beim Systemstart im LOG ausgegeben.

```
zwave: controller vendor: Aeon Labs, chip type: ZW0301, API version: 03.07, SDK version: 5.02 pl3
```

Z-Wave Komponenten müssen an dem nomos system angemeldet werden. Hierfür muss die Z-Wave Schnittstelle in den Include Mode versetzt werden. Der Include Mode ist für ca. 30s aktiv. In dieser Zeit muss dass entsprechende Z-Wave Device eingelernt werden. Dies erfolgt in der Regel durch das 3x Tasten der Mode- oder Programm Taste in einem kurzen Zeitabstand (1s). Die genaue Vorgehensweise ist der Gerätebeschreibung der jeweiligen Geräte zu entnehmen.

Ein erfolgreiches Anmelden der Z-Wave Komponenten an das nomos - System ist nur möglich, sofern die Komponente auch in der Z-Wave Datenbank vorhanden ist. Es besteht ebenfalls die Möglichkeit, eigene Datenbank Files zu erstellen und manuell in die Datenbank des nomos system einzutragen. Für diesen Fall stehen entsprechende Befehle zur Verfügung. Sollten Sie über eine Komponente verfügen die nicht erkannt wird, empfehlen wir jedoch den nomos system Support zu kontaktieren.

Die Z-Wave Datenbank des nomos systems basiert auf die Daten der Pepper One GmbH (http://www.pepper1.net), der Zertifizierungsstelle der Z-Wave Allianz. Das nomos system verfügt über einen direkten Zugriff auf die Aktualisierungen der Pepper One Datenbank. Es empfiehlt sich also vor dem Einlernen neuer Devices ein mögliches Update der Datenbank einzuleiten. Für diesen Zweck steht im Wizard eine entsprechende Möglichkeit zur Verfügung.



[1] Quelle: http://de.wikipedia.org/wiki/Z-Wave

Das Anmelden von Z-Wave Komponenten wird im weiteren Verlauf noch ausführlicher beschrieben.

Das Z-Wave AddOn benötigt ebenfalls eine Konfigurationsdatei. Je nach Vorgehensweise bei der Einrichtung kann diese Datei auch automatisch von dem nomos system angelegt werden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	zwave.csv
Länge:	1000 Zeilen/Einträge
Event:	-
Export:	-
Lizenz:	-

Die `zwave.csv` besteht aus 2 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[nodes]; Definition bzw. Namenszuordnung der Z-Wave Nodes (Devices)

Aufbau:

Im misc-Ordner liegt `zwave.csv` mit folgendem Inhalt (wird automatisch erzeugt):

- **[CONFIG];Sektion**

  ACTIVE;YES Aktiviert/Deaktiviert den Webserver
  „`YES`" (Default)= aktiv,
  „`NO`" = nicht aktiv

  DEBUG;YES Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

  PORT;{Portnummer} `"AUTO"` oder serieller Port, an dem der zwave Controller angeschlossen ist (`/dev/ttyS1, /dev/ttyUSB0` usw...), optional, default: `AUTO`

- **[NODES];Sektion** Zuordnung eines Klartextnamens zur Node ID. Die Node ID wird einmalig vom System vergeben, wenn ein Device eingelernt wurde.

  `{NodeIndex};{Nodename}`

Beispiele für die Definitionen [NODE] Sektion:

2;FEKO_TUER_WZ; Ordnet der `NODE_ID 2` den Namen „`FEKO_TUER_WZ`" zu

3;BWM_WZ; Ordnet der `NODE_ID 3` den Namen „`BWM_WZ`" zu

Die Node ID wird beim Einlernprozess von dem System automatisch erzeugt. Wurde ein Device erfolgreich eingelernt, wird automatisch ein Standard Eintrag (`x;NODE_x`) in der `zwave.csv` generiert, resp. die `zwave.csv` erzeugt. Der `{NodeIndex}` ist fest und darf nicht verändert werden. Der `{Nodename}` darf beliebig verändert werden.

Entfernen Sie keine Einträge aus der zwave.csv. Wenn Sie Geräte entfernen wollen, benutzen Sie bitte den EXCLUDE Befehl.

Einlernen und entfernen eines Z-WAVE Device/Node

Neue Z-Wave Devices müssen in das nomos system eingelernt werden. Um den Einlernprozess zu starten steht der Befehl <ZWAVE><INCLUDE={ON/OFF}><ZWAVE> zur Verfügung. Der Einlernprozess kann im Log verfolgt werden.

Wurde <ZWAVE><INCLUDE=ON><ZWAVE> ausgeführt, muss das einzulernende Device in den Einlernmodus versetzt werden. In der Regel wird dies durch Betätigung eines Tasters (3x kurz drücken) eingeleitet. Eine genaue Anweisung ist in der Betriebsanleitung zu den jeweiligen Geräten enthalten.

```
22:29:07.439 - parser: parsing sequence: <ZWAVE><INCLUDE=ON></ZWAVE>
22:29:07.439 - parser: command: ZWAVE - INCLUDE=ON
22:29:07.440 - parser: reply sequence: <ZWAVE>INCLUDE=|OK</ZWAVE>
22:29:07.469 - zwave: controller state: AddReady
```

Das erfolgreiche Einleiten des Einlernmodus wird vom System mit einem AddReady bestätigt. Nun muss das Z-Wave Device für das Einlernen aktiviert werden. Sobald der Controller das Z-Wave Device erkennt, wird dies im Log ausgegeben.

```
22:29:14.751 - zwave: controller state: AddNodeFound
22:29:14.933 - zwave: controller state: AddLearning
22:29:16.718 - zwave: controller state: AddDone
22:29:16.801 - zwave: controller state: Idle
22:29:17.692 - zwave: saving zwave data
```

Ist der Einlernprozess erfolgreich durchgeführt, ordnet der Controller dem Device eine ID zu. Ebenso wird die Konfigurationsdatei aus der Z-Wave Datenbank zugeordnet. Die zugeordnete ID sowie das zugehörige xml File wird im Log ausgegeben. Die zwave.csv wird um einen neuen Eintrag (11;NODE_11) ergänzt.

```
22:29:21.483 - zwave: new device NODE_11: Fibaro Motion Sensor (Multi-Sensor) file:
540-010f-0800-1001-03-03-43-02-04.xml (score: 9)
22:29:21.837 - zwave: saving zwave data
```

Möchten Sie ein weiteres Gerät einlernen, muss der Einlernprozess erneut gestartet werden.

Für das entfernen von Z-Wave Geräten stehen ebenfalls entsprechende Befehle zur Verfügung. Bitte entfernen Sie keine Einträge manuell aus der [NODE] Sektion der Z-Wave.csv. Sofern Sie ein vorhandenes Gerät aus der Konfiguration entfernen wollen, verwenden Sie den Befehl: <ZWAVE><EXCLUDE=ON><ZWAVE>. Anschließend versetzen Sie innerhalb von 30 Sekunden das zu entfernende Z- Wave Gerät, wie vor beschrieben, in den Einlernmodus.

```
09:57:28.242 - parser: command: ZWAVE - EXCLUDE=ON
09:57:28.243 - parser: reply sequence: <ZWAVE>EXCLUDE=|OK</ZWAVE>
09:57:28.271 - zwave: controller state: RemoveReady
09:57:42.208 - zwave: controller state: RemoveNodeFound
09:57:42.416 - zwave: controller state: RemoveLearning
09:57:42.417 - zwave: NODE_14 removed
09:57:42.428 - zwave: controller state: Idle
```

Wurde ein entsprechendes Z-Wave Device gefunden, wird auch der entsprechende Eintrag in der Controller Datenbank entfernt. Ebenso der Eintrag in der zwave.csv. Das Z-Wave Gerät selbst wird in den Auslieferungszustand versetzt.

Ist das zu entfernende Gerät defekt oder nicht mehr vorhanden, kann durch Verwendung des Befehls <ZWAVE><REMOVE={Node}><ZWAVE> das Entfernen von Geräten erzwungen werden.

Nachdem das System vollständig konfiguriert oder verändert wurde empfiehlt es sich, die Netzwerktopologie aktualisieren zu lassen. Diese Massnahme empfiehlt sich auch dann, wenn

die Örtlichkeiten der Z-Wave Devices verändert werden. Zu diesem Zweck steht der Befehl `<ZWAVE><UPDATENETWORK><ZWAVE>` zur Verfügung.

```
parser: parsing sequence: <ZWAVE><UPDATENETWORK></ZWAVE>\x0A
parser: command: ZWAVE - UPDATENETWORK
zwave: updating network topology
parser: reply sequence: <ZWAVE>UPDATENETWORK=|OK</ZWAVE>
```

Nach erfolgreicher Ausführung wird das Mesh Netzwerk neu erstellt bzw. aktualisiert.

Der Einlernprozess kann sich gelegentlich etwas mühsam gestalten. Werden die Geräte nicht erkannt, verringern Sie den Abstand zu Ihrem Z-Wave Controller und versuchen Sie es erneut.

Die BC Status Ausgabe

Alle Statusmeldungen werden (sofern vom Gerät unterstützt) auch ohne gezielte Abfrage im Broadcast gesendet. Dies betrifft auch Daten, für die es keine Befehle gibt (z.B. Batteriestatus, Temperatur, Luftfeuchtigkeit usw.) Bei batteriebetriebenen Geräte ist zu beachten, dass evtl. Konfigurationsänderungen erst nach dem nächsten (planmässigen oder manuellen) Wakeup des Gerätes übertragen und angewendet werden. Die Daten können mittels EventServer ausgewertet und verarbeitet werden.

Beim Systemstart werden alle Geräte einmalig gepollt. Im Log werden alle angemeldeten Nodes und deren Konfigurationsdateien ausgegeben. Die Ausgabe der Z-Wave Events erfolgt erst nach vollständigem Laden der Z-Wave Konfiguration.

```
zwave: engine started, library version: v1.5.0-rc3
zwave: NODE_2: Aeon Labs - Routing Binary Sensor file: 538-0086-0002-0036-03-03-43-01-05.xml
zwave: NODE_3: Fibar Group - Routing Binary Sensor file: 540-010f-0800-1001-03-03-43-02-04.xml
zwave: NODE_5: Everspring -  file: 203-011a-0101-0102-00-00-00-00-00.xml
zwave: NODE_6: Fibar Group -  file: 476-010f-0600-1000-03-03-34-16-16.xml
zwave: NODE_7: Aeon Labs - Routing Binary Sensor file: 538-0086-0002-0036-03-03-43-01-05.xml
```

Der Status der angemeldeten Geräte (binary-Switch und Mutilevel-Switch) wird automatisch im 60s Intervall gepollt. Die Pollzeiten der verschiedenen Datentypen werden ebenfalls im Log ausgegeben. Die Pollzeiten sind zZ nicht änderbar.

Sofern ein Device die Z-Wave Clock Klasse unterstützt wird die Uhrzeit automatisch im 12 Stunden Intervall gesetzt.

```
zwave: battery states will be updated every 60 minutes
zwave: metering data will be updated every 5 minutes
zwave: device clocks will be set every 12 hours
```

Ausgabe des Device Status im BC. Hierüber kann ebenfalls der Batteriestatus der jeweiligen Geräte über entsprechende EventServer Definitionen gematcht werden.

```
21:00:55.662 - bc: <ZWAVE><FEKO_TUER_WZ GENERAL PURPOSE=1></ZWAVE>
21:00:55.665 - bc: <ZWAVE><FEKO_TUER_WZ BATTERY=100 %></ZWAVE>
21:00:55.670 - bc: <ZWAVE><BMW_WZ GENERAL PURPOSE=0></ZWAVE>
21:00:55.674 - bc: <ZWAVE><BMW_WZ TEMPERATURE=25.40></ZWAVE>
21:00:55.686 - bc: <ZWAVE><BMW_WZ LUMINISCENCE=0.00></ZWAVE>

21:00:55.710 - bc: <ZWAVE><BMW_WZ BATTERY=100 %></ZWAVE>
21:00:55.752 - bc: <ZWAVE><SIREN=0></ZWAVE>

21:00:55.776 - bc: <ZWAVE><SIREN BATTERY=100 %></ZWAVE>
21:00:55.795 - bc: <ZWAVE><PLUG1=1></ZWAVE>
21:00:55.801 - bc: <ZWAVE><PLUG1 POWER=0.00></ZWAVE>
21:00:55.831 - bc: <ZWAVE><PLUG1 ELECTRIC=0.00></ZWAVE>
21:00:55.844 - bc: <ZWAVE><PLUG1 ELECTRIC=0.00></ZWAVE>
```

Bei Multilevel, Mehrkanal oder Multiple Instanzen wird im Broadcast der Device-Name mit "-`{x}`" ausgegeben bzw. ergänzt, sofern mehr als eine Instanz die Kommandoklasse unterstützt, z.B. bei einem 3Kanal RGB Dimmer mit dem Device Namen `<RGB_Leuchte>`:

```
21:00:55.776 - bc: <ZWAVE><RGB_LEUCHTE-1=255></ZWAVE>
21:00:55.795 - bc: <ZWAVE><RGB_LEUCHTE-2=25></ZWAVE>
21:00:55.801 - bc: <ZWAVE><RGB_LEUCHTE-3=125></ZWAVE>
```

Ebenfalls werden die Alarm Klassen unterstützt und gesondert ausgegeben. Bei jeglichen Alarm-Events erscheint im Broadcast `<ZWAVE><{Node} ALARM=1></ZWAVE>` bzw. bei der Rücknahme `<ZWAVE><{Node} ALARM=0></ZWAVE>`. Der Alarm Zustand der Geräte kann auch per Befehl zurückgesetzt werden.

```
21:00:55.746 - bc: <ZWAVE><BMW_WZ ALARM=1></ZWAVE>
```

Es existiert eine Befehlsklasse ZWAVE mit folgenden Befehlen:

Kommando	Argument	Beschreibung
Allgemein		
RESET		versetzt den zwave-Controller in den Auslieferungszustand und loescht alle angelernten Geraete
INCLUDE#	{ON/OFF}	startet/beendet den Include-Modus (Hinzufuegen von Geraeten)
EXCLUDE#	{ON/OFF}	startet/beendet den Exclude-Modus (Entfernen von Geraeten)
REMOVE#	{Node}	entfernt ein Geraet mit "Gewalt" (z.B. wenn das Geraet defekt ist und nicht mehr excluded werden kann)
DUMPDEVICES		gibt alle angelernten Geraete im Log aus
INTERVIEW#	{Node}	interviewt ein Geraet erneut (findet beim Include implizit statt)
LOADCONFIG#	{Node},{zwaveXML}	laedt eine andere zwave-XML aus der Library fuer das Geraet `{Node}`
SAVE		speichert den aktuellen Zustand des zwave Netzwerks auf der CF-Karte
UPDATENETWORK		aktualisiert die Z-Wave Netzwerk-Topologie, falls Z-Wave Nodes relokalisiert wurden
RESETALARM	{Node}	setzt alle Alarmmeldungen für das Gerät `{Node}` manuell zurück
Device Konfiguration		
CONFMODE#	{Port}	startet den zwave Stack im Konfigurationsmodus (keine `zwave.csv` erforderlich), gueltige Werte fuer `{Port}`: `/dev/ttyS1`, `/dev/ttyUSB0`, `AUTO`
CONFIGURE#	{Node}, {ConfOpt}, {ConfWert}	setzt die Konfigurationsoption `{ConfOpt}` des Geraetes `{Node}` auf `{ConfWert}` , `{ConfOpt}` ist ein Wert von 0-255
GETCONFIGURATION#	{Node}	liefert die Konfigurationswerte des Geraetes `{Node}`
Device Assoziation		
GETASSOCIATIONS#	{Node}	liefert alle Assoziationsgruppen des Geraetes `{Node}` und deren Inhalt
ADDASSOCIATION#	{Node},{Group}, {AddNode}	fuegt das Geraet `{AddNode}` in die Assoziationsgruppe `{Group}` auf dem Geraet `{Node}` ein, `{Group}` ist ein Index
REMOVEASSOCIATION#	{Node},{Group}, {RemoveNode}	entfernt das Geraet `{RemoveNode}` aus der Assoziationsgruppe `{Group}` auf dem Geraet `{Node}`
Wakeup Handling		

Kommando	Argument	Beschreibung
GETWAKEUPINTERVAL RANGE#	{Node}	liefert den fuer das Geraet {Node} gueltigen Wertebereich fuer den Wakeup (in Sekunden)
GETWAKEUPINTERVAL#	{Node}	liefert den aktuell eingestellten Wakeup-Wert fuer {Node} (in Sekunden)
SETWAKEUPINTERVAL#	{Node}	setzt den Wakeup-Wert fuer {Node} (in Sekunden)
Befehle		
SETSWITCH#	{Node}, [{Instanz/CH}], {Wert}	Schaltet den Schalter {Node} an bzw. aus, {Wert} kann sein: ON/OFF,0/1,YES/NO. Wird [{Instanz/CH}] verwendet kann auf mögliche weitere Instanzen (zB Mehrkanal Schalter) zugegriffen werden. Wird nur {Node} verwendet, wird eine Liste aller Instanzen des Gerätes ausgegeben.
GETSWITCH#	{Node}	liefert den Status des Schalters {Node}
SETLEVEL#	{Node}, [{Instanz/CH}], {Wert}	Setzt den Dimmer {Node} auf {Wert} oder schaltet {Node} ein bzw. aus, {Wert} kann sein: 0-99, ON/OFF, YES/NO. Sonst wie SETSWITCH#
GETLEVEL#	{Node}	liefert den Wert von {Node}
GETTHERMOSTATMODES#	{Node}	liefert alle unterstuetzten Modi des Thermostats {Node}
SETTHERMOSTATMODE#	{Node},{Mode}	schaltet den Thermostat {Node} auf {Mode} um
GETTHERMOSTATSET POINTS#	{Node}	liefert die Setpoints aller Modi des Thermostats {Node}
SETTHERMOSTATSET POINT#	{Node},{Mode},{Wert}	aendert den Setpoint des Thermostats {Node} im Modus {Mode} auf {Wert}
RESETMETER#	{Node}	setzt die aufgelaufenen Metering-Werte des Geraets {Node} auf 0

{Node} kann folgendes sein:

x Index des Gerätes (siehe Log)

NODE_x wie oben, nur mit Prefix NODE_

{Name} Der Name des Gerätes (aus der `zwave.csv`)

Beispiele:

<ZWAVE><SETSWITCH=NODE_11,1></ZWAVE> Schaltet den Schalter (Binärswitch) NODE_11 ein. .

<ZWAVE><SETLEVEL=Wohnen Dimmer,67></ZWAVE> Schaltet den Dimmer Wohnen Dimmer auf 67% .

<ZWAVE><DUMPDEVICES></ZWAVE> Gibt eine Liste aller bekannten Z-Wave Geräte zurück.

<ZWAVE><UPDATENETWORK></ZWAVE> Initiiert das Aktualisieren der Netzwerktopologie bzw. den Neuaufbau des Mesh Netzwerkes.

<ZWAVE><SETLEVEL=StripeRGB></ZWAVE> Liefert eine Liste aller Instanzen des Dimmers StripeRGB

<ZWAVE><SETLEVEL=StripeRGB,0,100></ZWAVE> Schaltet den Kanal 1 des Dimmers StripeRGB auf 100

<ZWAVE><SETSWITCH=NODE_34,1,OFF></ZWAVE> Schaltet den Kanal 2 des Schalters NODE_34 aus.

4.9 Philips HUE Support

Das nomos system unterstützt ebenfalls die HUE Leuchten der Fa. Philips. Die Erkennung der HUE-Bridge im Netzwerk erfolgt nach Betätigung der „Lern“ Taste der Bridge völlig automatisch. Danach wird die Kommandoklasse freigeschaltet. Es wird keine zus. Konfigurationsdatei benötigt. Im LOG kann man erkennen, ob eine HUE Bridge erkannt wird. Sofern eine HUE Bridge vorhanden, werden auch die spezifischen Daten der Bridge ausgegeben. Ebenso gibt das Log Auskunft über die an der HUE Bridge angemeldeten Leuchten

```
hue: detected 'Philips hue bridge 2012', model number: 929000226503, serial number: 00178815217b @
192.168.50.12
hue: class 'HUE' enabled
hue: connected lights:
hue: light id: 1, name: 1_Hue Lamp, type: Extended color light, modelid: LCT001, swversion:
66009663, reachable: false1
```

Das Konfigurieren Ihres HUE Systems bzw. das Anmelden der einzelnen Leuchten entnehmen Sie bitte der Gebrauchsanleitung des HUE Systems.

Technische Daten:

- Max. Anzahl Lampen: 50 pro Bridge
- Zigbee Light link: Protokoll 1.0 zertifiziert
- Frequenzbereich: 2400-2483,5 MHz
- Es ist eine Building Automation Lizenz (BAOS/HS/KNX) erforderlich

Die BC Status Ausgabe

Der Status der Leuchten wird alle 5 Sekunden gepollt und bei Änderungen im Broadcast ausgegeben. Die Ausgabe erfolgt auch dann, wenn die Leuchten über die HUE eigene App angesteuert werden. Diese Meldungen können mittels entsprechenden EventServer ausgewertet und weiterverarbeitet werden.

```
bc: <HUE>3_Buero=1|ID=3|BRIGHTNESS=69|HUE=36952|SATURATION=253|COLORTEMP=153|REACHABLE=1|OK</HUE>
```

Gruppen Support

Das nomos system unterstützt auch Gruppenfunktionen. Das Anlegen einer HUE-Gruppe erzeugt eine neue, “virtuelle” Lampe mit dem Namen `{group}`. Der Status wird sofort als Broadcast gesendet. HUE Gruppen können genau wie eine einzelne Lampe mit den bekannten Befehlen gesteuert werden, die Steuerung und die Feedbacks sind völlig transparent und einheitlich. Eine Gruppe wird im Broadcast immer als reachable/erreichbar ausgegeben. Nachfolgend ein Auszug der Broadcast Ausgabe der Gruppe „`G1`“ mit den zugehörigen Leuchten „`2_Stehl.`“ und „`3_Buero`“

```
bc: <HUE>2_Stehl.=0|ID=2|BRIGHTNESS=224|HUE=47124|SATURATION=253|COLORTEMP=500|REACHABLE=1|OK</HUE>
bc: <HUE>3_Buero=0|ID=3|BRIGHTNESS=224|HUE=40375|SATURATION=252|COLORTEMP=500|REACHABLE=1|OK</HUE>
bc: <HUE>G1=0|ID=1|BRIGHTNESS=224|HUE=40375|SATURATION=252|COLORTEMP=500|REACHABLE=1|OK</HUE>
```

Bei der Bildung von Gruppen ist bei der Namensgebung zu beachten, dass ein bereits vergebener Name für eine Leuchte nicht als Gruppenname verwendet werden sollte. Ist dies jedoch der Fall, wird nicht die Gruppe sondern die Leuchte angesteuert.

Es existiert eine Befehlsklasse HUE mit folgenden Befehlen:

Kommando	Argument	Beschreibung
GETALLLIGHTS		liefert eine Liste aller verbundenen Leuchten-Namen
GETALLLIGHTIDS		liefert eine Liste aller verbundenen Leuchten-IDs

Kommando	Argument	Beschreibung
RENAME#	{Light},{Name}	benennt {Light} in {Name} um
POWER_ON#	{Light}	schaltet {Light} ein
POWER_OFF#	{Light}	schaltet {Light} aus
SETBRIGHTNESS#	{Light},{Wert}	aendert die Helligkeit von {Light} auf {Wert} - {Wert} darf 0..255 sein
SETHUE#	{Light},{Wert}	aendert den Farbton von {Light} auf {Wert} - {Wert} darf 0 - 65535 sein
SETSATURATION#	{Light},{Wert}	aendert die Farbsaettigung von {Light} auf {Wert} - {Wert} darf 0 - 255 sein
SETCOLORTEMP#	{Light},{Wert}	aendert die Farbtemperatur von {Light} auf {Wert} - {Wert} haengt ab von den angesteuerten Leuchten und darf z.Zt. von 153 - 500 sein (6500K - 2000K)
SETEFFECT#	{Light}, COLORLOOP/NONE	fuehrt auf {Light} eine ColorLoop aus oder beendet sie
SETALERT#	{Light},SELECT/ LSELECT/NONE	fuehrt auf {Light} einen (SELECT) oder mehrere (LSELECT - für 30 Sekunden) "Breathe-Cycle" aus oder beendet sie (NONE)
CREATEGROUP	{group}, {light1}, {light2},..., {lightx}	erzeugt eine neue Gruppe mit dem Namen {group} und den nachfolgenden Lampen {light1}, {light2}...
DELETEGROUP	{group}	löscht die Gruppe {group}
GETALLGROUPS		liefert alle in der HUE-Bridge konfigurierten Gruppen
GETGROUPLIGHTS	{group}	liefert eine Liste aller der Gruppe {group} zugeordneten Lampen

{Light} kann eine Leuchten-ID oder der Name der Leuchte sein.

{GROUP} ist immer ein Gruppenname.

Beispiele:

<HUE><POWER_ON=2></HUE><HUE><POWER_OFF=2></HUE> Schaltet die HUE Leuchte mit der „ID 2" ein oder aus.

<HUE><SETBRIGHTNESS=BUERO,128></HUE> Schaltet die HUE Leuchte mit dem zugeordneten Namen BUERO auf ca 50% ein.

<HUE><SETHUE=BUERO,643></HUE> Schaltet die HUE Leuchte mit dem zugeordneten Namen BUERO in einen Rotton.

<HUE><CREATEGROUP=G1,2,3></HUE> Erstellt die Gruppe „G1" mit den zugehörigen Leuchten „ID 1" und „ID 2"

<HUE><POWER_OFF=G1></HUE> Schaltet die HUE Gruppe mit den Namen „G1" aus.

<HUE><DELETEGROUP=MyGroup><CREATEGROUP=MyGroup,BUERO,STEHL.></HUE> Das Ändern einer Gruppe kann innerhalb einer Sequenz erfolgen. Hierbei wird zuerst die Gruppe gelöscht und im zweiten Schritt die Gruppe erneut angelegt

4.10 mremote Support (commandFusion iViewer support)

Die mremote Schnittstelle erweitert die nomos Infrastruktur mit dem `Control System Protocol` um eine weitere Kommunikationsschnittstelle. Das verwendete Protokoll ist speziell für den Betrieb in einer Infrastruktur mit geringer Bandbreite, wie es zB in Wireless- oder Mobilfunknetzen üblich ist, optimiert. Die Umsetzung bzw. die Verwendung dieses bidirektionalen* Protokolls erfolgt im Hintergrund.

**Gegenstelle führt nicht nur aus, sondern bestätigt auch die Durchführung der Anfrage*

Mit mremote ist es ebenfalls möglich, unter Verwendung der iViewer Software , eine frei definierbare, grafisch gestützte Benutzeroberfläche (GUI) für mobile Apple und/oder Android Devices in die nomos Infrastruktur zu integrieren. Über diese Oberfläche können beliebige Aktionen aus dem gesamten Portfolio der nomos Dienste/Befehle abgerufen und fernbedient werden. Darüber hinaus transcodiert nomos das iViewer Projekt in HTML5. Somit kann die GUI auch unter Verwendung von HTML5 fähigen Browsern auf nahezu jedem Endgerät verwendet werden. Für die Verwendung der HTML5 GUI auf IOS oder Android Geräten ist ebenfalls eine APP verfügbar. - App Store - Play Store

Informationen zur Einrichtung bzw. dem Umgang mit der iViewer Software finden Sie auf der Homepage des Herstellers. Hinweise wie das nomos system die iViewer Projektdaten und Lizenzdateien verwaltet, finden Sie in dem Kapitel Mini- Webserver.

Um den HTML5 Server auf einen Mac nutzen zu können, muss zus. MAMP auf dem Mac System installiert sein. Bei den rein Linux basierten Systemen ist der HTML5 Server vorinstalliert. Für den Betrieb des HTML5 Servers ist der mremote Dienst zwingend erforderlich.

Beim Einsatz mehrerer nomos systeme im Netzwerk kann durch die Anwendung der Relay-Funktion ein Mehrfachzugriff realisiert werden. Hierüber kann die Kommunikation gezielt auf ein Zielsystem „gelenkt" werden. Dies bedeutet, dass mit nur einer GUI-Oberfläche eine Steuerung aller Systeme innerhalb des Netzwerkes erfolgen kann, sofern funktional die Joins auf den verschiedenen Systemen mit identischen Funktionen belegt wurden. Ebenso ist es möglich mit mehreren Remote Clients auf verschiedene Systeme, auch parallel, zuzugreifen.

Das Control System Protocol unterstützt lediglich die Verwendung und das Übertragen von drei Variablen (JOINS) Typen, die den Elementen in der GUI zugeordnet werden. Diese JOINS gliedern sich wie folgt:

digital Join binäre Elemente, wie zB Schalter/Taster

analog Join 2Byte Zahlenwerte für zB Slider

seriell Joins ASCII für zB die Übertragung von Textinhalten, URL's oder ähnliches

Ebenso können Array's (List Joins) aus den jeweiligen Typen generiert und übertragen werden.

Die JOINS und deren Funktionen müssen nun dem `nomos` System mitgeteilt werden, damit eine Kommunikation bzw. ein Datenaustausch erfolgen kann. Die gesamte Definition der JOINS und deren Funktionalität erfolgt über eine Definitionsdatei, der `mremote.csv`. Sind die JOINS definiert, können diese auch intern über entsprechende Befehle manipuliert werden. Ebenso können die JOINS zB auch direkt in dem Logik Modul verwendet werden (s. Kapitel Logik).

Definierte Joins bzw. deren Wertinhalt werden automatisch gepuffert und nach erfolgter Verbindung an den Client übertragen. Dieses Verhalten kann bei Bedarf je JOIN unterdrückt werden (NOINIT).

Verhalten der JOINS

Wie bereits beschrieben, wird der Wertinhalt der JOINS automatisch gepuffert und bei einem Client connect automatisch übertragen. Wird der Wertinhalt eines JOINS über das `Control System Protocol` verändert, werden entsprechend definierte JOIN Actions ausgeführt. Das Ändern der JOIN

Inhalte über die nomos system Befehle dient nur der Aktualisierung der JOINS auf dem Client, führen also **keine** Aktionen aus.

Die für die Aktualisierung der JOINS definierbaren Events aktualisieren ebenfalls nur die JOINS auf dem Client und führen keine definierten Aktionen aus. Sollen Events auch Aktionen ausführen, müssen diese in der `{Local-Action}` definiert werden. Bei digitalen JOINS gilt dies nur für den `{Push-Event}` wenn die Bedingung „WAHR" ist

Die `mremote.csv` Datei

Die `mremote.csv` befindet sich im Ordner `.\misc` des nomos Projektverzeichnisses. Diese Datei ist für die Kommunikation zwischen der iViewer App oder der nomos App und die Verwendung des HTML5 Servers zwingend erforderlich. Die `mremote.csv` beschreibt die gesamte Funktionalität der eigentlichen GUI Anwendung und bildet das Kommunikationsgatway zur nomos Infrastruktur..

Die `mremote.csv` Definitionen erlauben Möglichkeiten, wie zB die automatische Skalierung analoger Werte, Feedback Definitionen, Vergleicher, „push and hold" Funktionalität, u.v.m. Auf Seiten des Clients muss dies also nicht gesondert berücksichtigt werden. Bei Verwendung der iViewer Software können also sämtliche Logiken oder auch Funktionen zu Aktualisierung der jeweiligen Zustände vernachlässigt werden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	mremote.csv
Länge:	10000 Zeilen/Einträge, 100 LIST Definitionen
Event:	BC, Status
Export:	-
Lizenz:	ja

Die `mremote.csv` besteht aus 6 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[DIGITAL]; Definition der digitalen JOINS

[ANALOG]; Definition der analogen JOINS

[SERIAL]; Definition der seriellen JOINS (bis 8K Textinhalt je JOIN)

[LIST={aJOIN}]; Definition der Liste mit dem Index x (x entsprechend der Analog-Join Nummer für die Liste im GUI-Designer) (max. 250 Elemente/Liste)

[PAGE]; Hier kann man auf den vom iViewer bzw. HTML5-Server gesendeten Namen der Seite matchen, die ein Client aufruft und eine beliebige Aktion ausführen.

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES Aktiviert/Deaktiviert den Webserver
  „`YES`" (Default)= aktiv,
  „`NO`" = nicht aktiv

  DEBUG;YES Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

PORT;{Port} Portnummer, für eingehende Verbindungen `5500` (Default)

PASSWORD;{Passwort} Definition eines Passwortes, dass vom mremote-Client beim Verbindungsaufbau erwartet wird

PORTRAIT;{Action} nomos Sequenz, die beim Drehen des iPods/iPhones in die vertikale Position ausgeführt wird

LANDSCAPE;{Action} nomos Sequenz, die beim Drehen des iPods/iPhones in die horizontale Position ausgeführt wird

CONNECT;{Action} nomos Sequenz, die beim Verbindungsaufbau eines mremote Clients ausgeführt wird

GUI_IP;{IP,AUTO} Server-IP oder Host name, die beim Projekt Download über den integrierten Webserver automatisch an den Client übergeben wird. Bei `AUTO` wird automatisch die lokale IP eingetragen

GUI_PORT;{Port}/AUTO Server-Port, der beim Projekt Download über den integrierten Webserver automatisch an den Client übergeben wird.Bei `AUTO` wird automatisch die unter PORT definierte Portnummer eingetragen. `1235` (Default)

SPEECHLANGUAGE;{Lang-Setting} wählt die Sprache fuer die Sprachausgabe für die Device Steuerung.
(s Kapitel Device Control). `en_US` (Default:)

SPEECHVOICE;{Voice} wählt die Stimme für die Sprachausgabe für die Device Steuerung.
(s Kapitel Device Control)

Eine Liste der möglichen Sprachen und Stimmen gibt es hier: http://dragonmobile.nuancemobiledeveloper.com//public/index.php?task=supportedLanguages

- **[DIGITAL];Sektion**

  {Nummer};{Typ};{Push-Action};{Release-Action};{Push-Event};{Release-Event};{Option};{Local-Action}

  Aufbau der Definition eines digitalen Joins

  Erläuterungen:

  {Nummer} Nummer/Adresse des digitalen JOIN (dJOIN)

  {Typ} Typ des Buttons, mögliche Werte:

  0,1,# normales JOIN Verhalten. Bei Einsatz von „`#`" wird der eingehende Wert gepuffert und kann mittels „`\#`" der Action übergeben werden.

  TOGGLE* Button rastet ein.

  INTERVAL={x,y} führt `{Push-Action}` sofort aus und nach y (Anfangsverzögerung in ms) die `{Release-Action}`. Die `{Release-Action}` wird bei gedrücktem Button in der eingestellten Intervall Zeit (x in ms) wiederholt. Wird (y) nicht angegeben, ist y=0.

  XOR={x}* weist den Button einer Gruppe x von Buttons zu, die gegeneinander verriegelt sind. x muß eine Zahl sein. Bei Verwendung dieses Typs, wird automatisch die `{Option}NOCACHE` gesetzt.

HOLD={x} führt `{Push-Action}` aus, wenn der Button weniger als x ms gedrückt gehalten wird. Anderenfalls wird nach Ablauf von x ms sofort die `{Release-Action}` ausgeführt

{Push-Action} Script oder nomos Sequenz, die beim Drücken einer Taste ausgeführt wird

{Release-Action} Script oder nomos Sequenz, die beim Lösen einer Taste ausgeführt wird

{Push-Event} nomos Match-Sequenz, die den Button in den „pushed“- Zustand versetzt. Ist die Matchbedingung erfüllt, wird ebenfalls die `{Local-Action}` ausgeführt.

{Release-Event} nomos Match-Sequenz, die den Button in den „released“- Zustand versetzt.

{Option} Zusätzliche Konfigurationsoptionen, mögliche Werte:
(Es können auch mehrere Optionen gesetzt werden.
Mehrere Optionen können durch „,“ getrennt werden)

NOCACHE Erzwingt die Aktualisierung eines JOINS. Andernfalls erfolgt eine Aktualisierung nur bei Zustandsänderung.

NOINIT Unterdrückt das automatische aktualisieren eines JOINS bei einem Client Connect

AUTOFEEDBACK Sorgt dafür, dass der vom JOIN gesendete Wert sofort wieder an den Join zurückgeschrieben wird. Von der Funktion her also identisch zu dem “simulate feedback” Flag des iViewers, jedoch mit dem Unterschied, dass alle verbundenen Clients sofort aktualisiert werden.

DELAYFEEDBACK=x Blockiert das “echte” Feedback (also das Match) eines JOINS so lange, bis nach der letzten(!) Aktion des Joins x Millisekunden vergangen sind.
Zwischenzeitlich eintreffende Matches werden automatisch zwischengespeichert und nach Ablauf der Zeit wird der letzte Wert an den Join herausgeschrieben. Darduch wird verhindert, dass `AUTOFEEDBACK` und ein richtiges Match miteinander kollidieren.

{Local-Action} Script oder nomos Sequenz, die bei einem erfolgreichen Match zusätzlich lokal ausgeführt wird. Bei Listen ist neben dem Standard-Argument-Platzhalter `\#` zusätzlich der Listen-interne-Index `\&` möglich.
Die `Local-Action` eines Joins wird auch dann ausgeführt, wenn ein ggfs. definiertes `DELAYFEEDBACK` noch nicht abgelaufen ist.

*Das Simulate Feedback Flag darf bei Verwendung dieser Option nicht gesetzt sein

- **[ANALOG];Sektion**

{Nummer};{Bedingung};{Skalierung};{Action};{Event};{Option};{Local-Action} Aufbau der Definition eines analogen Joins

Erläuterungen:

{Nummer} Nummer/Adresse des analogen JOIN (aJOIN)

{Bedingung} Match Bedingung, die das Event-Argument erfüllen muß.
Der eingehende Wert wird gepuffert und kann mittels „`\#`“ der Action übergeben

werden.
Mögliche Werte:

{leer} oder # Wildcard-Match (immer erfüllt)

{<>=Wert} Argument muß kleiner/größer/gleich {Wert} sein, Kombination Vergleichsoperatoren ist möglich

{Skalierung} rechnet den maximalen Zahlenwert eines JOINS (65535) auf `{Skalierung}` (=Zahlenwert) um. Die Umrechnung erfolgt bidirektional. {Skalierung} kann auch eine nomos Match-Sequenz sein wie zB `\%GETCTIME=\*|` , um die Skalierung dynamisch zur Laufzeit anzupassen.

{Action} Script oder nomos Sequenz, die bei eingehenden JOIN Aktivitäten ausgeführt werden soll. Der Inhalt des Joins kann mit `\#` als Argument eingefügt bzw. übergeben werden.

{Event} nomos Match-Sequenz, mit der der JOIN synchronisiert werden soll. Die aus dem Eventserver bekannten Argument-Filter wie zB `\*, \?, \%` u.v.m. sind auch hier verwendbar.

{Option} Zusätzliche Konfigurationsoptionen, mögliche Werte:
(Es können auch mehrere Optionen gesetzt werden.
Mehrere Optionen können durch „,“ getrennt werden)

NOCACHE Erzwingt die Aktualisierung eines JOINS. Andernfalls erfolgt eine Aktualisierung nur bei Zustandsänderung.

NOINIT Unterdrückt das automatische aktualisieren eines JOINS bei einem Client Connect

AUTOFEEDBACK Sorgt dafür, dass der vom JOIN gesendete Wert sofort wieder an den Join zurückgeschrieben wird. Von der Funktion her also identisch zu dem "simulate feedback" Flag des iViewers, jedoch mit dem Unterschied, dass alle verbundenen Clients sofort aktualisiert werden.

DELAYFEEDBACK=x Blockiert das "echte" Feedback (also das Match) eines JOINS so lange, bis nach der letzten(!) Aktion des Joins x Millisekunden vergangen sind.
Zwischenzeitlich eintreffende Matches werden automatisch zwischengespeichert und nach Ablauf der Zeit wird der letzte Wert an den Join herausgeschrieben. Darduch wird verhindert, dass `AUTOFEEDBACK` und ein richtiges Match miteinander kollidieren.

ACTIONINTERVAL=x Erzwingt einen Mindestabstand (`x in ms`) beim Auslösen einer Action einzuhalten. Während dieser Wartezeit eintreffende Events werden ignoriert. Nur der letzte Wert der Events wird gespeichert und nach Ablauf der Wartezeit die Action mit dem letzten Wert als Argument ausgeführt.

{Local-Action} Script oder nomos Sequenz, die bei einem erfolgreichen Match zusätzlich lokal ausgeführt wird. Bei Listen ist neben dem Standard-Argument-Platzhalter `\#` zusätzlich der Listen-interne-Index `\&` möglich.
Die `Local-Action` eines Joins wird auch dann ausgeführt, wenn ein ggfs. definiertes `DELAYFEEDBACK` noch nicht abgelaufen ist.

- **[SERIAL];Sektion**

{Nummer};{Bedingung};{Action};{Event};{Option};{Local-Action} Aufbau der Definition eines seriellen Joins

Erläuterungen:

{Nummer} Nummer/Adresse des seriellen JOIN (sJOIN)

{Bedingung} Match Bedingung, die das Event-Argument erfüllen muß.
Der eingehende Wert wird gepuffert und kann mittels „`\#`" der Action übergeben werden.
Mögliche Werte:

- **{leer} oder #** Wildcard-Match (immer erfüllt)
- **{String}** Argument muß identisch zu `{String}` sein
- **{<>=Wert}** Argument muß kleiner/größer/gleich {Wert} sein, Kombination Vergleichsoperatoren ist möglich

{Action} Script oder nomos Sequenz, die bei eingehenden JOIN Aktivitäten ausgeführt werden soll. Der Inhalt des Joins kann mit `\#` als Argument eingefügt bzw. übergeben werden.

{Event} nomos Match-Sequenz, mit der der JOIN synchronisiert werden soll. Die aus dem Eventserver bekannten Argument-Filter wie zB `\*, \?, \%` u.v.m. sind auch hier verwendbar.

{Option} Zusätzliche Konfigurationsoptionen, mögliche Werte:
(Es können auch mehrere Optionen gesetzt werden.
Mehrere Optionen können durch „," getrennt werden)

- **NOCACHE** Erzwingt die Aktualisierung eines JOINS. Andernfalls erfolgt eine Aktualisierung nur bei Zustandsänderung.
- **NOINIT** Unterdrückt das automatische aktualisieren eines JOINS bei einem Client Connect
- **AUTOFEEDBACK** Sorgt dafür, dass der vom JOIN gesendete Wert sofort wieder an den Join zurückgeschrieben wird. Von der Funktion her also identisch zu dem "simulate feedback" Flag des iViewers, jedoch mit dem Unterschied, dass alle verbundenen Clients sofort aktualisiert werden.
- **DELAYFEEDBACK=x** Blockiert das "echte" Feedback (also das Match) eines JOINS so lange, bis nach der letzten(!) Aktion des Joins x Millisekunden vergangen sind.
  Zwischenzeitlich eintreffende Matches werden automatisch zwischengespeichert und nach Ablauf der Zeit wird der letzte Wert an den Join herausgeschrieben. Darduch wird verhindert, dass `AUTOFEEDBACK` und ein richtiges Match miteinander kollidieren.
- **ACTIONINTERVAL=x** Erzwingt einen Mindestabstand `(x in ms)` beim Auslösen einer Action einzuhalten. Während dieser Wartezeit eintreffende Events werden ignoriert. Nur der letzte Wert der Events wird gespeichert und nach Ablauf der Wartezeit die Action mit dem letzten Wert als Argument ausgeführt.

{Local-Action} Script oder nomos Sequenz, die bei einem erfolgreichen Match zusätzlich lokal ausgeführt wird. Bei Listen ist neben dem Standard-Argument-Platzhalter `\#` zusätzlich der Listen-interne-Index `\&` möglich.

Die `Local-Action` eines Joins wird auch dann ausgeführt, wenn ein ggfs. definiertes `DELAYFEEDBACK` noch nicht abgelaufen ist.

- **[LIST={x}];Sektion**

  {Join-Typ=Nummer};{Master-Join-Typ=Nummer};{Event};{Optionen};{Local-Action}
  Aufbau der Definition eines List Joins

  Erläuterungen:

  {Join-Typ=Nummer} Beschreibt den Join, für den List-Support gewünscht wird, z.B. `SERIAL=7` oder `ANALOG=3` oder `DIGITAL=23`, sinnvoll ist eigentlich nur `SERIAL`.

  {Master-Join-Typ=Nummer} Beschreibt den Join, an den der o.g. Join angehängt werden soll.d.h. wenn der Master-Join aus der Liste ein Event schickt, wird ein Event für den o.g. Join simuliert, Syntax wie oben

  {Event} die Match-Sequenz zur Erzeugung eines Listeneintrages, der Match erfolgt subtraktiv.

  {Optionen} z.Zt. nur `NOINIT` unterstützt

  {Local-Action} nomos Sequenz, die ausgeführt werden soll, wenn ein Match erfolgt. Hier ist als Wildcard neben dem Argument `\#` zusätzlich `\&` für den Listen-Index möglich

- **[PAGE];Sektion**

  {Page Match};{Action}; Aufbau der Definition eines Page Event

  Erläuterungen:

  {Page Match} Name der Page, die auf dem Client aufgerufen wird

  {Action} Script oder nomos Sequenz, die bei einem erfolgreichen Page Match ausgeführt werden soll. Der Inhalt des Page Match kann mit `\#` als Argument eingefügt bzw. übergeben werden.

Beispiele für die Definitionen [DIGITAL] Sektion:

{Nummer};{Typ};{Push-Action};{Release-Action};{Push-Event};{Release-Event};{Option};{Local-Action}

10;;;;;;NOCACHE; Registriert einen JOIN (10) ohne weitere Funktion. Diese Möglichkeit dient zB als Platzhalter oder zum Zwischenspeichern von möglichen JOIN Werten, die mittels den
`<SETDIGITALJOIN=` Befehlen entsprechend zugeweisen werden können.

2;;;;0/2/12-%=1>;0/2/12-\%=0>;NOCACHE;<SCRIPT><RUN=home.myh></SCRIPT>
Definiert einen JOIN (2) für eine binäre Anzeige. Die Anzeige reagiert auf den Event einer KNX Gruppenadresse. In diesem Beispiel auf die Adresse 0/2/12. Ist die Bedingung für den `{Push-Event}` „WAHR", wird ebenfalls die `{Local-Action}` ausgeführt und das Script `home.myh` gestartet. Diese Art der Definition kann zB verwendet werden, wie ein reiner Event Server.

15;1;<DENON><POWERON></ITUNES>; Die einfachste Definition eines JOIN. Hier wird der JOIN (15) mit einer einfachen Funktion belegt, wie es zB bei IR gesteuerten Geräten der Fall ist.

66;TOGGLE;<SYS><MUTEON></SYS>;<SYS><MUTEOFF></SYS>;|MUTE=ON|;MUTE=OFF| ;NOINIT;<SYS><SETSERIALJOIN=10,Mute></SYS>

Eine Definition eines JOINS (66), die bei Verwendung der iViewer GUI einen einrastenden Button erzeugt. Ebenfalls sind 2 Funktionen hinterlegt, die bei einem Zustandswechsel entsprechend ausgeführt werden. Der Statuts wird automatisch durch entsprechende Events aktualisiert. Ist die Bedingung für den `{Push-Event}` „WAHR“ , wird ebenfalls die `{Local-Action}` ausgeführt und der sJOIN mit dem Text „Mute“ beschrieben. Die Funktion ist beispielhaft für die Mute Steuerung auf einem Apple Mac Client.

54;HOLD=800;<SYS><DNKEY></SYS>;<SYS><PUSHDNKEY></SYS>; Definiert den JOIN (54) mit einer Funktion, die die `{Push-Action}` ausführt, wenn der Button kürzer als die hold time (ms) gedrückt wird. Wird der Button länger als die eingestellte hold time gedrückt, wird nach Ablauf die `{Relase-Action}` sofort ausgeführt, auch wenn der Button gedrückt gehalten wird.

18;INTERVAL=500;;<SYS><VOLUP=[STEP]></SYS>; Definiert den JOIN (18) mit einer Funktion, die bei gedrücktem Button im Intervall von 500ms die `{Push-Action}` ausführt. Die Funktion ist beispielhaft für die Steuerung einer Lautstärke, die bei gedrückter Taste langsam erhöht werden soll. Die Schrittweite wurde in diesem Beispiel per Variable übergeben. Der Einsatz von Variablen bietet die Möglichkeit, die Schrittweite zur Laufzeit dynamisch ändern zu können.

10;XOR=10;<ITUNES><PLAY></ITUNES>;;GETPLAYERSTATE=PLAYING;;NOCACHE
11;XOR=10;<ITUNES><PAUSE></ITUNES>;;GETPLAYERSTATE=PAUSED;;NOCACHE
12;XOR=10;<ITUNES><STOP></ITUNES>;;GETPLAYERSTATE=STOPPED;;NOCACHE

Definition einer Gruppe (10) von sich selbst verriegelnden Buttons (10,11,12). Die Funktion synchronisiert den Zustand alle Buttons einer XOR Gruppe. Es kann hierbei nur ein Button den Pushed Zustand annehmen. Wird der entsprechende Button gedrückt, löst dies auch die `{Push-Action}` des zugehörigen `JOINS` aus. Das Aktualisieren der Buttons kann auch per `{Push-Event}` erfolgen.

Beispiele für die Definitionen [ANALOG] Sektion:

{Nummer};{Bedingung};{Skalierung};{Action};{Event};{Option};{Local Action}

1;#;100;<SYS><VOLSET=\#></SYS>;|VOL=*|; Empfängt den analogen JOIN 1 und skaliert den empfangenen Wert auf „100“. Der skalierte Wert wird an den Befehl der `{Action}` übergeben. Die `{Event}` Definition aktualisiert den JOIN sofern ein entsprechender Wert empfangen wird. Dies ist das typische Besspiel für zB einen Volume Slider

3341;#;255;<KNX><SETVALUE=1/2/3,\#></KNX>;1/2/9-\%=*>;NOCACHE, ACTIONINTERVAL=300, DELAYFEEDBACK=800;;

Funktion, wie vor beschrieben. Hier werden jedoch zus. Optionen definiert. Durch die `NOCACHE` Option wird der JOIN bei jedem eingehenden Wert aktualisiert, egal ob eine Wertänderung erfolgt oder nicht. Mit `ACTIONINTERVAL` wurde ein Sendepuffer aktiviert, der die `{Action}` nur alle 300ms ausführen lässt. `DELAYFEEDBACK` verzögert die Aktualisierung der Anzeige auf den Client, die durch `{Event}` angetriggert wird. Die Verzögerungszeit beginnt erst nachdem keine Wertänderungen mehr über den JOIN eingehen.

Dies ist eine KNX typische Dimmer/Slider Funktion. Hier ist es insbesondere wichtig, dass die Kommunikationsgeschwindigkeit des mremote Dienstes auf die Geschwindigkeit des Zielsystems angepasst werden kann.

12600;#;100;<NUVO><SET_ID=16><ID_VOL(V)=\#></NUVO>;Z16_VOLUME=*|;NOCACHE, ACTIONINTERVAL=300,DELAYFEEDBACK=800;

Funktion wie vor, jedoch mit einer Skalierung von 100 für die Steuerung einer Lautstärke für ein NUVO Multiroomsystem.

30001;#;<[ATV-1]>\%GETCTIME=*|;;<[ATV-1]>\%GETPTIME=*|; Funktion für eine dynamische Anzeige. Die `{Skalierung}` der Anzeige wird über ein Event Match permanent angepasst. Dieses Beispiel dient der Anzeige der Laufzeit von Musiktiteln. Da die Laufzeit der jeweiligen Titel unterschiedlich ist, muss die Skalierung entsprechend angepasst werden.

Beispiele für die Definitionen [SERIAL] Sektion:

{Nummer};{Bedingung};{Action};{Event};{Option};{Local Action}

2;#;;|GETTITLE=*| Beschreibt den JOIN (2) mit dem aktuellen iTunes Titel

2152;;;2/3/13-\%=*>;NOCACHE;; Beschreibt den JOIN (2152) mit dem Wertinhalt der KNX Gruppenadresse 2/3/13. Durch die NOCACHE Option wird der JOIN auch dann aktualisiert, wenn keine Wertänderung erfolgt. Dieses Beispiel ist typisch für zB eine KNX Temperaturanzeige.

50000;#;<SCRIPT><RUN=PUSH(txt).myh><ARG=\#></SCRIPT><SYS><SETSERIALJOIN=50000,> </SYS>

Diese Funktion empfängt einen Text über den JOIN (50000) und überträgt diesen mittels „`\#`“ an die `{Action}`. Eine weitere Funkton beschreibt/löscht diesen JOIN wieder. Diese Funktion ist beispielhaft für das Übertragen von Textmitteilungen.

50001;Password;<SCRIPT><RUN=ALLOFF.myh></SCRIPT> Diese Funktion empfängt einen Text über den JOIN (50001) und vergleicht den Text mit dem Inhalt der `{Bedingung}`. Ist das Ergebnis „WAHR“ wird die `{Action}` ausgeführt.

30005;;;<[ATV-1]>\%GETPTIME{%02m:%02s}=*|;; Diese Funktion beschreibt den JOIN (30005) mit einem Wert. Der Wert wird von GETPLAYTIME empfangen und in der Form umgewandelt, dass die Anzeige in Minuten und Sekunden ausgegeben wird. Für die Kommando Klasse wurde hier eine Systemvariable verwendet.

210;;;<HS>\%<100/0/8\%=*;;<SYS><SETSERIALJOIN=210,\#{%*1.00} A></SYS>; Diese Funktion empfängt einen `{Event}` für die Adresse `100/0/8` von dem KO-Gateway des GIRA Homeservers. Sobald eine Wertänderung der Daten erkannt wird, wird der gematchte Wert an die `{Local Action}` übergeben und ausgeführt. Der Wert wird durch die Funktion `{%*1.00}` auf 2 Nachkommastellen formatiert. Zusätzlich wird noch eine Einheit „`A`“ angehangen

30002;;;<[ATV-1]>\%|TITLE=*|;NOCACHE;<SYS><SETSERIALJOIN=30000, http://[SYSTEM_IP]:[WEBSERVPORT]/[ATV-1]_current.jpg></SYS>;

Diese Funktion beschreibt den JOIN (30002) mit einem Wert. Der Wert wird von `TITLE` empfangen. Immer wenn der `{Event}` Daten empfängt, wird auch die `{Local Action}` ausgeführt. Diese Funktion ist beispielhaft für das Empfangen einer Titelinformation von zB AppleTV. Bei jedem Empfang der Titelin-

formation wird auch die aktuelle URL des Covers übermittelt. Auch hier wurde die Kommando Klasse mittels Systemvariable übergeben. Zus. wurde auch mit fixen Variablen für die Übertragung der Netzwerkdaten (IP und Portadresse) des Webservers übertragen.

Weitere Informationen im Umgang mit Systemvariablen entnehmen Sie bitte dem Kapitel Arbeiten mit Systemvariablen

Beispiele für die Definitionen [LIST=n] Sektion:

Bei Listen mit mehreren JOINS wird die Liste nur dann gelöscht, wenn das Event für die 1. Join-Definition innerhalb der Liste positiv ist. Andernfalls werden die Inhalte der List nur überschrieben.

{Join-Typ=Nummer};{Master-Join-Typ=Nummer};{Event};{Optionen};{Local-Action}

Erzeugen einer Liste aller iTunes-Playlisten:

[SERIAL];91;#;<ITUNES><PLAYPLAYLIST=\#></ITUNES>;GETALLPLAYLISTS=*OK; Die obige Matchsequenz für einen seriellen JOIN liefert eine Liste aller Playlisten: `Mediathek|Musik|....|zuletzt hinzugefügt|`

[LIST=1];SERIAL=91;DIGITAL=90;*|;; In der List-Definiton wird auf diese Liste (1) mit `\*|` subtraktiv gematcht und damit die Liste gefüllt. Gleichzeitig wird das virtuelle Array des Serial Join 91 an den Digital Join 90 gebunden, der im GUI-Designer als Listenelement definiert wurde.

Beispiele für die Definitionen [PAGE] Sektion:

{Page Match};{Action};

Home;<SYS><SETSERIALJOIN=17,Welcome Home></SYS> Wird die Page „Home" auf dem Client geöffnet, wird die {Action} ausgeführt

Subpage_*;<SYS><SAY=You are at Subpage \#></SYS> Matcht den Namen der Subpage und übergibt den Wert an die {Action}

Relay Funktion

Sie haben außerdem die Möglichkeit, die mremote-Steuerung per RELAY an weitere nomos-Systeme umzuleiten. Hierzu tragen sie bitte als nomos Sequenz ein:

RELAY={IP}:{Port},{MAC};

Aufbau der Definition einer RELAY Definition in der Sektion [DIGITAL]

Erläuterungen:

{IP} IP des entfernten mremote Dienst, wohin die Kommunikation geroutet werden soll.

{Port} Port des entfernten mremote Dienst, wohin die Kommunikation geroutet werden soll.

{MAC} `{MAC}` des entfernten mremote Dienst, wohin die Kommunikation geroutet werden soll. Wird die MAC angegeben, wird ein WOL-Paket versendet, bevor das `RELAY` aufgebaut wird.

Wird die `RELAY` -Funktion als Action ausgelöst, baut der lokale Mac eine Verbindung zu einem entfernten Mac auf. Sobald die Verbindung aufgebaut ist, werden alle Actions und Events transparent weitergeleitet. Die lokalen JOIN-Definitionen haben in diesem Modus keine Wirkung, lediglich die RELAY-Actions werden noch lokal evaluiert. Der lokale Mac verhält sich in diesem Modus völlig transparent.

Diese Technologie verwendet man zB bei Systemen, die nahezu gleich konfiguriert sind. Somit kann man eine einheitliche GUI für alle zu steuernde nomos systeme generieren. Die nomos systeme können dann gezielt durch entsprechenden Aufbau des `RELAY's` angesprochen werden,

Beispiele für die RELAY Definitionen:

{Page Match};{Action};

530;1;RELAY=192.168.1.23:5500,C82A1400D30; Öffnet das `RELAY` auf dem entfernten nomos system. Es kann immer nur eine `RELAY` Verbindung aktiv sein. Wird bei einer bestehenden `RELAY` Verbindung erneut ein `RELAY` aufgebaut, so wird die bestehende Verbindung automatisch geschlossen.

531;1;RELAY=OFF; Schließt das `RELAY` und aktiviert die lokalen `JOIN` Definitionen

Die `RELAY`-Funktion kann nicht zusammen mit nomos-Sequenzen kombiniert werden.

Für die Manipulation der JOINS, stehen folgende <SYS> Befehle zur Verfügung:

Kommando	Argument	Beschreibung
SETDIGITALJOIN#	{Nummer},{Wert}	Weist den JOIN {Nummer} einen neuen {Wert} zu.
SETANALOGJOIN#	{Nummer},{Wert}	Weist den JOIN {Nummer} einen neuen {Wert} zu.
SETSERIALJOIN#	{Nummer},{Wert}	Weist den JOIN {Nummer} einen neuen {Wert} zu.

Beispiele:

<SYS><SETDIGITALJOIN=156,1></SYS> Setzt den digitalen JOIN 146 auf den Wert „1“. Dieser Befehl wirkt sich nicht auf Aktionen der JOIN Definitionen aus und aktualisiert lediglich den JOIN auf dem Client.

<SYS><SETANALOGJOIN=50,44></SYS> Wie vor, jedoch wird hier der analoge JOIN 50 mit dem Wert 44 beschrieben.

<SYS><SETSERIALJOIN=100,Power ON></SYS> Wie vor, jedoch wird hier der serielle JOIN 100 mit dem Text „Power ON“ beschrieben.

KAPITEL 5

Sonstiges

5.1 Arbeiten mit Systemvariablen

Das nomos system unterstützt die Verwendung von Systemvariablen. Sie können unter Verwendung der `...VAR`-Befehle aus der `<SYS>` Klasse beliebige Werte speichern oder gespeicherte Werte setzen und auch verwenden. Dies ist bei grösseren Installationen sinnvoll, da hierüber z.B. auch Script Aufrufe manipuliert, Lautstärken oder Systemstatus definiert und gespeichert werden können. Neben der Verwendbarkeit in dem Logik Modul lassen sich auch Matches, Klassen oder Kommandos systemweit über Sysvars definieren, die zur Laufzeit substituiert werden. Sysvars können also nahezu überall verwendet werden.

Sysvars können zur Laufzeit in eine Datei `sysvars.csv` und/oder flüchtig im Speicher geschrieben werden.

Bei den Linux Versionen ist der Name der Datei fixiert. Bei Mac Systemen kann die `sysvars.csv` unter Verwendung der folgenden Option in der PrefPane unbenannt werden. Wird mit der optionalen Definitionsdatei gearbeitet, liegt diese dann auch im Root Verzeichnis des entsprechenden Projektes. Wird diese Datei nicht gefunden bzw. ist sie nicht vorhanden, wird automatisch die Default Datei verwendet. Die Verwendung dieser Methode dient zB dazu, individuelle Systemkonfigurationen über verschiedene Dateien zu steuern. Änderungen können in die optionale Datei geschrieben werden. Sollen die Standardwerte wieder verwendet werden, muss lediglich die optionale Datei entfernt werden, oder die Checkbox in der PrefPane deaktiviert werden.

☐ Try to use sysvars.csv in project root for system variables

Das Schreiben bzw. Überschreiben der Einträge in die `sysvars.csv` kann auch unterbunden werden (s. Befehl `LOCKVARS=ON/OFF`). In diesem Fall werden Änderungen und Neueinträge nicht explizit auf den Datenträger geschrieben, sondern flüchtig im Speicher gehalten. Nach einem Neustart werden bei `LOCKVARS=ON` nur die Werte aus der Datei eingelesen. Flüchtige Werte gehen verloren. Dieses Verhalten kann während der Laufzeit verändert werden. Ein `LOCKVARS=ON` schreibt nicht den aktuellen Inhalt des flüchtigen Speichers in die Datei!

Ein `LOCKVARS=ON` schreibt nicht den aktuellen Inhalt des flüchtigen Speichers in die Datei! Insbesondere bei Benutzung von CF oder SD Karten, wie in den nomos Box Linux Varianten verwendet, sollte man ein permanentes Beschrieben des Datenträgers verhindern. Die Default Einstellung ist bei den Linux Versionen daher auch `LOCKVARS=ON`.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\

Name:	sysvars.csv (wird automatisch generiert, falls nicht vorhanden)
Name opt.:	Wie in der PrefPane konfiguriert (nur Mac OSX).
Länge:	10000 Zeilen/Einträge
Event:	-
Export:	-
Lizenz:	-

Die `sysvar.csv` enthält keine Sektionen

Die Werte der Variablen werden automatisch eingesetzt, wenn in der Kommandosequenz der String `[{Variablenname}]` gefunden wird.

Aufbau der Datei: {Name};{Inhalt};

Erläuterungen:

{Name} Name der Variable, der Abruf erfolgt dann entsprechend `[{Name}]`

{Inhalt} Wertinhalt der Variable, durch welchen der Platzhalter `[{Name}]` ersetzt wird.

Beispiele:

<SYS><SETVAR=STARTVOLUME,33></SYS> Speichert eine Variable Namens „Startvolume“ mit dem Wert 33

<SYS><SETVOL=[STARTVOLUME]></SYS> Setzt die Systemlautstärke auf den Wert der Variable „Startvolume“

<SYS><GETALLVARS></SYS> Liefert die Ausgabe aller vorhandenen Variablen und deren Wertinhalt.

Dynamische Systemvariablen:

Es werden sog. dynamische Systemvariablen automatisch generiert. Diese Variablen werden bei Systemstart definiert und dienen zB der automatischen Übergabe systemspezifischer Parameter wie Bildschirmauflösung, IP Adresse, URL des Projektpfades und vieles mehr.

[SYSTEM_IP]	- aktuelle IP
[SYSTEM_SERIAL]	- aktuelle Seriennummer der Hardware
[SYSTEM_MAC]	- aktuelle MAC Adresse der Hardware
[SCREEN_X]	- x-Auflösung des Bildschirms
[SCREEN_Y]	- y-Auflösung des Bildschirms
[BUTTON_SIZE]	- aktuelle Höhe der Buttonleiste
[HOMEDIR]	- aktuelles Verzeichnis
[PROFILE]	- aktuelle Profilname (Kurzname)
[PROJECT]	- aktueller Projektpfad
[TIME]	- aktuelle Systemzeit (im KNX EIS3 Format)
[DATE]	- aktuelles Systemdatum (im KNX EIS3 Format)
[OEM_NAME]	- aktueller OEM Name des Demons
[ARG]	- Reserviert für die Argument Übergabe (s. Kapitel Script)
[GARG]	- Reserviert für die Argument Übergabe (s. Kapitel Script)

Sämtliche Variablen können überschrieben werden, d.h. eine manuelle Definition geht vor. Der Befehl

GETALLVARS liefert immer alle Variablen zurück, d.h. wenn z.B. SYSTEM_IP in der .csv definiert ist, kommt die Variable zweimal in der Antwortsequenz vor.

Durch die Einbindung der dynamischen Systemvariablen steht ein mächtiges Werkzeug zur Verfügung um Profile dynamisch aufzubauen. Nahezu sämtliche Einstellungen können mittels Systemvariablen übergeben werden. Auch CONFIG-Einstellungen/Sektionen der jeweiligen Definitionsdateien, wie z.B. IP- oder Port- Adressen können mittels Systemvariablen übergeben werden. Somit ist es möglich spezifische Einstellungen der Konfiguration allein aus der sysvars.csv zu steuern.

Änderungen der Variablen während der Laufzeit müssen per SETVAR -Befehl erfolgen.

Beispiel (webserver.csv):

[CONFIG];
ACTIVE;YES
DEBUG;[DEBUG]
PORT;1234
GUIPORT;1235
DVD_PATH;[HOMEDIR]/Movies/

Durch den Einsatz der dynamischen Variable [HOMEDIR] wird der Paramete DVD_PATH automatisch mit der URL des aktuell angemeldeten Users um den Eintrag „/Movies" vervollständigt und übergeben. Dieser Eintrag kann selbstverständlich auch direkt in der webserver.csv vorgenommen werden, jedoch würde es den Zweck einer vererbbarer Konfigurationsdatei verfehlen.

Die Option für den Debug Modus ist ebenfalls über eine Variable gesetzt. So kann man gezielt und vor allem systemweit diesen Mode programmatisch aktivieren oder deaktivieren

Systemvariablen können sich auch automatisch wie Feldvariablen verhalten, wenn die Wertinhalte durch ein „|" getrennt werden.

Ein gezieltes Schreiben auf einen Index ist z.Z. nicht möglich!

In der Befehlsklasse SYS existieren folgende Befehle für die Verwendung von Sysvars:

Kommando	Argument	Beschreibung
GETVAR=	{Variablenname}	liest den Wert einer System-Variablen
SETVAR=	{Variablenname}, {Wert}	ändert den Wert einer System-Variablen bzw. legt sie an
DELVAR=	{Variablenname}	löscht eine System-Variable
GETALLVARS		Liefert alle definierten Variablen incl. deren Wertinhalt
LOCKVARS=	{ON/OFF}	Verriegelt oder ermöglicht das Schreiben der Variablen auf den Datenträger
WAITVAR=	{Variablenname}, {Wert}{,{Timeout}}	Hält die aktuelle Sequenz so lange an, bis die Sysvar {Variablenname} den Wert {Wert} angenommen hat. Wird zusätzlich {Timeout} (in ms) angegeben, so wird {Timeout} - Millisekunden gewartet. Wenn die Zeit abgelaufen ist und {Variable} niemals {Wert} hatte, wird die restliche Sequenz NICHT abgearbeitet, sondern ignoriert. Hiermit lässt sich ein zeitabhängiges "IF -> THEN" nachbilden.

Beispiele:

<SYS><SETVAR=TEST,Wert1|Wert2|Wert3|Wert4></SYS> Hierüber wird nun eine Variable „`Test`“ mit den Inhalten „`Wert1`“, „`Wert2`“, „`Wert3`“, „`Wert4`“ in die sysvars.csv geschrieben. Das Trennzeichen „|“ indiziert hierbei den Index des jeweiligen Werteintrages.

<SYS><GETVAR{%l2,1}=Test></SYS> Die Abfrage mit einem **Formatparameter** soll nun den Wert des zweiten Eintrages der Variable „`Test`“ mit einer Länge von 1, also lediglich einen Eintrag, aus der vorherig angelegten Variablen liefern.
Das Ergebnis lautet entsprechend:
<SYS>GETVAR{%l2,1}=Wert2|OK</SYS>

<SYS><GETVAR{%l1,2}=Test></SYS> Die Abfrage wie gehabt, jedoch mit einer Länge von 2.
Das Ergebnis lautet entsprechend:
<SYS>GETVAR{%l1,2}=Wert1|Wert2|OK</SYS>

<SYS><WAITVAR=MESSAGE,HELLO,3000><SAY=Message was Hello></SYS> Führt den `SAY`-Befehl nur dann aus, wenn die Sysvar `MESSAGE` innerhalb von `3` Sekunden (oder bereits beim Aufruf von `WAITVAR`) den Inhalt `"HELLO"` hatte.

5.2 Mini- Webserver

Mini- Webserver (Command Fusion Extension)

Im nomos system ist ein kleiner Webserver eingebunden worden, dessen Aufgabe es ist, Informationsinhalte bereitzustellen. Diese Inhalte (Content) können Systemen von Drittanbietern zur Verfügung gestellt werden. So wird z.B. immer das Cover des aktuell angespielten iTunes-Song unter einer statischen Web-Adresse bereitgestellt. Ebenso können Bildinformationen des aktuellen TV-Senders (Sender Logo) oder das Cover der aktuell abgespielten DVD abgerufen werden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	webserver.csv
Länge:	100 Zeilen/Einträge (Aliasse)
Event:	
Export:	-
Lizenz:	-

Die `webserver.csv` **besteht aus 2 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[ALIASES]; Definition von Datenpunkten, die über KNXnet/IP empfangen und ausgewertet werden sollen.

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES Aktiviert/Deaktiviert den Webserver
  „`YES`" (Default)= aktiv,
  „`NO`" = nicht aktiv

  DEBUG;YES Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

  PORT;{Portnummer} Port des Webservers. Für `{Portnummer}` sollte man einen Port > 1024 (Default) verwenden.

  GUIPORT;{Portnummer} Port für den GUI Upload (iViewer) des Webservers. Für `{Portnummer}` sollte man PORT + 1 (Default=1235) verwenden, sofern verfügbar.

  DVD_COVER;{Bildname} Legt den DVD Covernamen fest (Default: `preview.jpg`)

  DVD_PATH;{Pfad zu den DVD-Verzeichnissen} Startpfad des DVD Verzeichnisses. Z.B „`DVD_PATH;/ Users/mmh/Movies/`" für den lokalen Ordner „Filme"

- **[ALIASES];Sektion** Der Webserver versucht alle Anfragen, denen `"alias/"` voransteht, mit Hilfe der `webserver.csv` in eine andere URL umzuwandeln. Es sind max. 100 Einträge möglich.

  {Aliasname};{URL} Name des Alias; zugeordnete URL

Beispiele für die Definitionen [ALIASES] Sektion:

– **Statisch:** `mein_bild;http://www.mremote.de/Start_files/droppedImage.png`
Aufruf: `http://{mmh-ip}:{Webserver-Port}/alias/mein_bild`

– **Mit fest definierter Sysvar:** `Bild_1;[BILD_LINK_1]`
Aufruf: `http://{mmh-ip}:{Webserver-Port}/alias/Bild_1`
Ersetzt Bild_1 durch die URL, die zuvor in der sysvar BILD_LINK_1 abgespeichert wurde. Die Evaluierung der sysvar erfolgt zur Laufzeit.

– **Mit Matching und mehreren Sysvars:** `Bild_\*;[START][BILD_LINK_\#]`
Aufruf wie oben, jedoch dynamisch: Bild_x wird durch die sysvar BILD_LINK_x ersetzt, sofern definiert. Das Argument wird eingefügt, bevor die sysvar ersetzt wird.

– **Mit Matching, ohne Sysvars:** `Bild_\*;http://www.mremote.de/Image_\#`
Aufruf wie oben, jedoch wird der Index des Bildes in eine andere URL eingebaut.

Anwendung:

– **iTunes Cover Support** *Folgende URL's werden von dem Webserver bereit gestellt:*

`http://{IP}:{Portnummer}/itunes_current.jpg` Zeigt das Cover des laufenden iTunes Titels

`http://{IP}:{Portnummer}/itunes_ID_{ID}.jpg` Zeigt das Cover des iTunes Titels nach Aufruf der ID

`http://{IP}:{Portnummer}/itunes_album_{Name}.jpg` Zeigt das Cover des iTunes Album nach Aufruf des Album Namens.

– **Remote (AppleTV) und SONOS Support** *Folgende URL's werden von dem Webserver bereit gestellt:*

`http://{IP}:{Portnummer}/{Class}_current.jpg` Zeigt das Cover des laufenden Titels der verwendeten Klasse, wie in der `remote.csv` oder `sonos.csv` definiert.

– **XBMC Support** *Folgende URL's werden von dem Webserver bereit gestellt:*

`http://{IP}:{Portnummer}/{Class}_current.jpg` Zeigt das Cover des laufenden Titels der verwendeten Klasse, wie in der `xbmc.csv` definiert.

`http://{IP}:{Portnummer}/{Class}_ ALBUMID_{AlbumID}.jpg`
Zeigt das Cover der Album ID des Players der verwendeten Klasse, wie in der `xbmc.csv` definiert.

– **DVD-Cover Support (DVD Player.app):** *Folgende Aufrufe sind möglich:*

`http://{IP-Adresse}:{Portnummer}/dvd_current.jpg` Liest den aktuellen DVDTitel aus und sucht in `{DVD_PATH}/{DVDTitel}` nach einer Datei `preview.jpg` und liefert diese zurück. `{DVDTitel}` wird hierbei aus den Informationen der laufenden DVD automatisch generiert.

`http://{IP-Adresse}:{Portnummer}/dvd_{`DVDTitel`}.jpg` Sucht in `{DVD_PATH}/{`DVDTitel`}` nach einer Datei `preview.jpg` und liefert diese zurück.

Beispiel: `http://192.168.50.55:1234/dvd_Casino%20Royale.jpg`
html-Zeichen (`%`) werden automatisch konvertiert. `preview.jpg` kann beliebig tief im zu durchsuchenden Ordner liegen

- **TV-Logo Support (eyeTV.app):** Im misc-Ordner einen Unterordner "web" anlegen. In diesem Ordner müssen die Senderlogos als *.png oder *.jpg abgelegt werden Es ist auf folgende Namenskonvention zu achten: tv_{Sendername}.jpg. {Sendername} muss dabei exakt dem Namen entsprechen, der von dem Befehl GETCHNAME der Befehlsklasse <TV> zurückgeliefert wird.

  Beispiel: Logo Dateien tv_SAT.1.jpg oder tv_SAT.1.png für den Sender SAT1 Unterverzeichnis „./misc/web" anlegen.

  Die Logos lassen sich durch Angabe des Dateinamens vom Webserver herunterladen um z.B. eine Programmliste zu erstellen. Ausserdem gibt es auch hier die fest definierte tv_current.jpg.

  Der Aufruf: http://{IP-Adresse}:{Portnummer}/tv_current.jpg oder / tv_current.png liest den aktuellen Sender aus und liefert das entsprechende Logo (wenn vorhanden).

  Für alle Grafiken gibt es sog. Dummys. Diese werden ausgegeben, wenn keine entsprechende Grafikdatei gefunden wird. Diese Dateien müssen ebenfalls im ./misc/web Ordner abgelegt und wie folgt benannt werden: itunes_dummy.jpg, dvd_dummy.jpg, tv_dummy.jpg {AppleTV}_dummy.jpg {sonos}_dummy.jpg oder eben als .png Datei.

  Groß- und Kleinschreibung spielt weder bei den Dateinamen noch bei den URL-Aufrufen eine Rolle.Der Grafikpuffer wurde auf maximal 20MB definiert. Damit sollten sich alle Grafiken darstellen lassen. Es sollte jedoch darauf geachtet werden, je nach Anwendung, die Dateien klein zu halten und nur für die Darstellung notwendige Pixelgrößen zu verwenden.

- **Support von freien Bilddateien:** Die gewünschten Dateien müssen im Ordner ./misc/web abgelegt werden. Dort abgelegte Dateien dürfen nicht mit den Features des Webservers kollidieren. Eine ITUNES_CURRENT.jpg im web Ordner würde z.b. nie ausgeliefert werden, wenn eine remote-Klasse mit dem Namen ITUNES existiert.

  ```
  http://{IP-Adresse}:{Portnummer}/{Dateiname}.jpg
  http://{IP-Adresse}:{Portnummer}/{Dateiname}.jpg
  ```

  Liefert die unter der URL abgelegte Bilddatei aus.

  Beispiel: Die gewünschte Datei (1.OG-Gang.jpg) muss im Ordner ./misc/web abgelegt werden. Das Zielsystem hat die IP Adresse 192.168.50.19 und der Webserver ist auf Port 1234 erreichbar. Der korrekte Aufruf lautet dann wie folgt:

  http://192.168.50.19:1234/1.OG-Gang.jpg

Bei Aufrufen mit '/' im Dateinamen wird der '/' durch ein BLANK ersetzt. Dies kann zb bei TV oder Musiktiteln der Fall sein.

5.2.1 Mini- Webserver (Command Fusion Extension)

Über den Mini- Webserver erfolgt auch die Bereitstellung der Projektdaten der iViewer Software (GUI Designer Projektdaten). Der iViewer und der GUI Designer ist ein Produkt von CommandFusion. Die Anwendung wird vollumfänglich vom nomos system unterstützt.

Informationen zur Einrichtung bzw. dem Umgang mit der iViewer Software finden Sie auf der iViewer Homepage des Herstellers. Hinweise zur Einrichtung bzw. zur Verknüpfung mit dem nomos system finden Sie im Kapitel mremote Support (commandFusion iViewer Support).

Darüber hinaus transcodiert das nomos system das iViewer Projekt in HTML5. Somit kann die GUI auch unter Verwendung von HTML5 fähigen Browsern auf nahezu jedem Endgerät verwendet werden. Für die Verwendung der HTML5 GUI auf IOS oder Android Geräten ist ebenfalls eine APP verfügbar. - App Store - Play Store

Um den HTML5 Server auf einem Mac nutzen zu können, muss zus. MAMP auf dem Mac System installiert sein. Bei den rein Linux basierten Systemen ist der HTML5 Server vorinstalliert. Für den Betrieb des HTML5 Servers ist der mremote Dienst zwingend erforderlich.

Datenablage (Projektfile)

Der gesamte Inhalt des GUI-Designer-Projektverzeichnisses (also Projektdatei `*.gui`, `*.asv` und die gesamten Bilddateien) müssen auf das nomos system kopiert werden. Es werden auch Unterverzeichnisse unterstützt, sofern diese in der Projektstruktur vorhanden sind. Das Zielverzeichnis für die Projektdaten auf dem nomos system lautet:

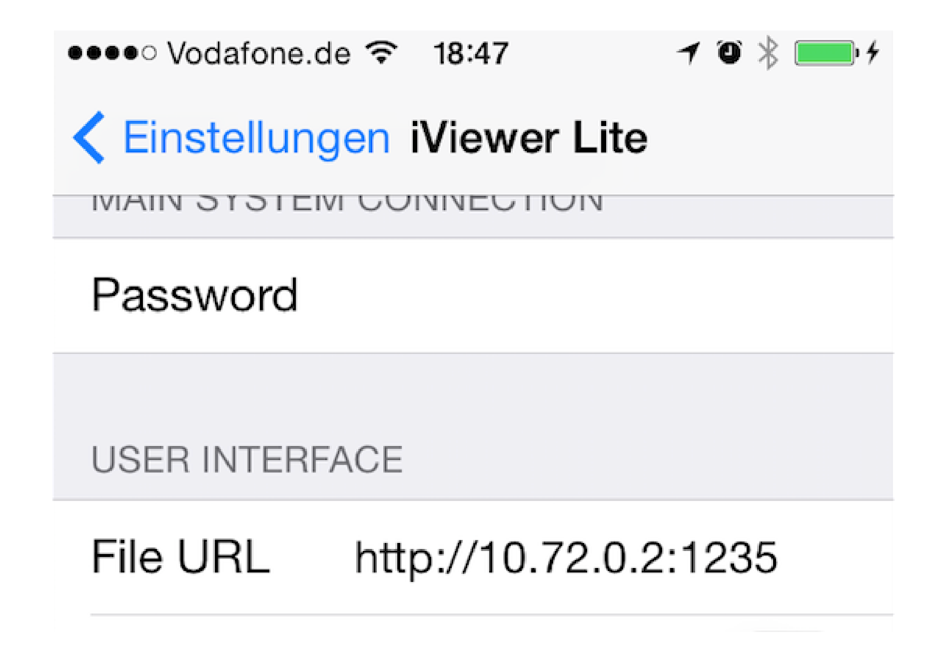
> .\{projectPath}\misc\mremote\

iViewer Upload Service

Der iViewer Upload Service wird über den Mini- Webserver bereitgestellt. Die maximale GUI Upload Größe beträgt 20MB.

Sind die Daten auf dem nomos system abgelegt worden, muss die `File URL` in den Einstellungen der entsprechenden APP eingetragen werden. Die `File URL` stellt sich wie folgt zusammen:

> *http://{IP-nomos system}:{Portnummer}/{Projectfile}.gui*
>
> > Die `{Portnummer}` für den Download Service ist, sofern unter `GUIPORT` in den Einstellungen der `webserver.csv` nicht anders definiert, immer +1 des Ports des Mini- Webservers. Bei Verwendung der Standardwerte wäre die Portadresse somit `1235`. Ist nur ein .gui File in der Ordnerstruktur abgelegt worden, ist die Angabe des Projektfiles nicht notwendig.



Bei dem Download der Projektdaten werden auch automatisch die Control System Settings übertragen, sofern dies nicht im `.gui` File vorhanden sind. Sollen nicht automatisch die Daten des Upload Servers übernommen werden, können diese in der `mremote.csv` in der `[CONFIG]` Sektion auch individuell vorgegeben werden.

Die CommandFusion iViewer Pro Versionen benötigen ebenfalls ein zus. Lizenzfile. Die Vorgehensweise zum Erwerb einer solchen Lizenz finden Sie auf der Homepage des Herstellers.

Die Geräte Lizenz, die Sie nach der Registrierung vom Hersteller erhalten, enthält folgende Informationen:

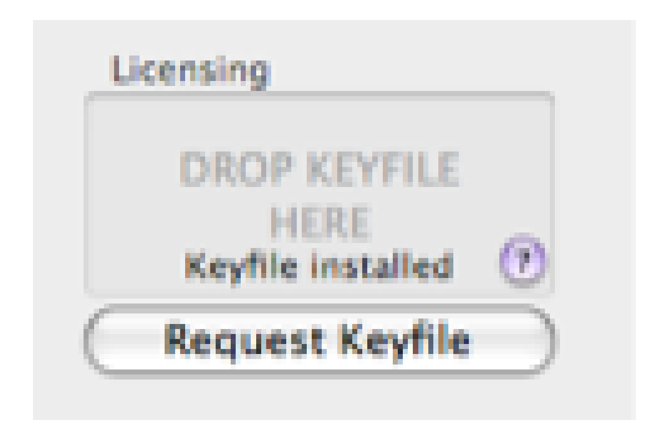
> Device Name: DMTABGTHFKYC
> Device ID: C9F80881-B67B-43B4-B272-4GF04FFF55D2
> License code: 567A5575D1830BD2EFD0621DB7C53095

Damit diese Daten nun ebenfalls bei dem Projekt Upload übertragen werden können, gehen Sie wie folgt vor:

Mac OSX

Erstellen Sie eine `.txt` Datei mit den Registrierungsinformationen. Beachten Sie bitte, dass die jeweiligen Zeilen an deren Ende keine Leerzeichen enthalten sondern unmittelbar mit einem LF (Line Feed) oder CR (Carriage Return) abgeschlossen wurden. Den Filenamen der .txt Datei können Sie frei vergeben. Bei älteren Versionen kann es notwendig sein, dass sie `License code:` in `Registration Code:` ändern müssen.

Öffnen Sie die nomos system Konfiguration (PrefPane) in den OSX Systemeinstellungen und ziehen Sie das .txt File per Drag & Drop auf das „Licensing“ Feld und folgen Sie dem Dialog.



Sie können beliebig viele Keyfiles an dem nomos system anmelden.

Linux Systeme

Bei den Linux basierten Systemen werden die Registrierungsinformationen in einer Textdatei (`remote_licenses.csv`) zusammengefasst. Diese Datei befindet sich in dem Ordner `./config`. Ist die datei nicht vorhanden, muss diese entsprechend angelegt werden. Je Zeile werden die Informationen wie folgt eingetragen:

Device Name;Device ID;License code

Beispielhaft:

DMTABGTHFKYC;C9F8765381-B67B-43B4-B272-4GF04FFF55D2;877A5575D1830AB563FD0621DB7C53095
DMTABGTAVGYC;C9F823451-B67B-43B4-B272-4GF04FFF55D2;907A5575D1830AS56FD0621DB7C53095
DMTABGTHGH4YC;C9CF9971-B67B-43B4-B272-4GF04FFF55D2;447A5575D1830B674FD0621DB7C53095

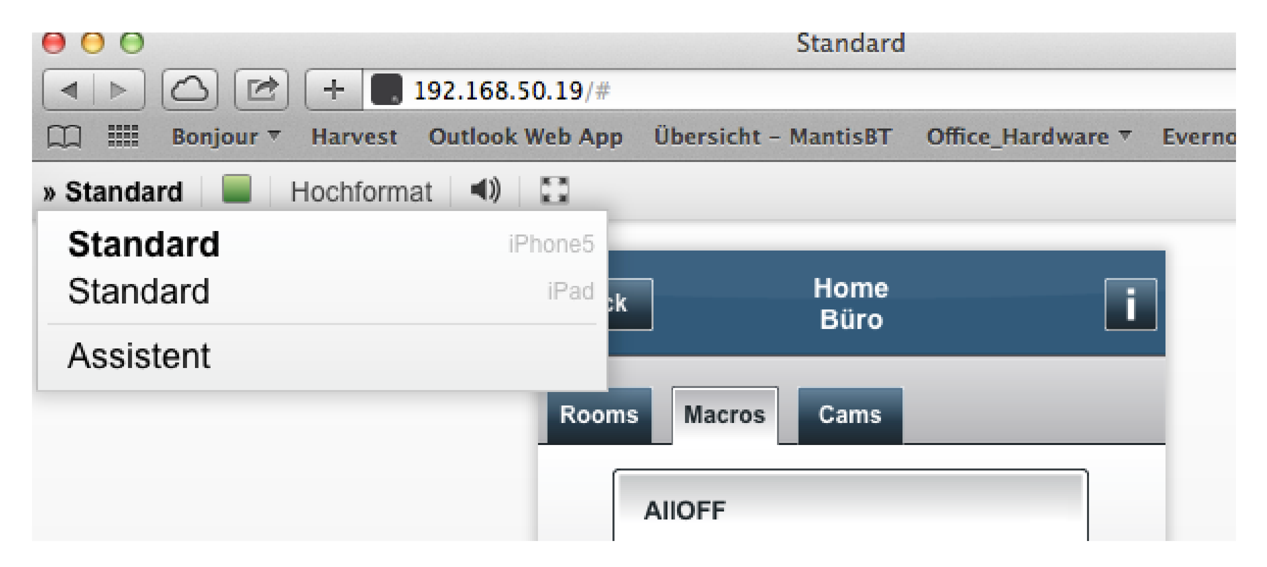
HTML5 Transcodierer

Wie bereits beschrieben, wird das iViewer Projektfile automatisch in HTML5 transcodiert und kann somit über jeden HTML5 fähigen Browser abgerufen werden. Bitte beachten Sie, dass es je nach Verwendung des Browser zu unterschiedlichen Darstellungen kommen kann. Wir empfehlen die Verwendung des Safari Browsers.

Unter HTML5 ist keine zusätzliche Client Lizenz erforderlich. Bitte beachten Sie auch, dass unter Umständen die Darstellung nicht 1:1 gleich der Darstellung im iViewer ist. Insbesondere ist bei den verwendeten Texten darauf zu achten, dass die verwendeten Fonds auch von dem eingesetzten Browser unterstützt werden. Gesten werden auch nur teilweise umgesetzt. Bei Listen ist darauf zu achten, dass die Elementgrößen exakt passend ausgelegt werden. Andernfalls könnten unerwünschte Scroll Balken erzeugt werden. Auch JavaScript, sofern verwendet, wird vollständig umgesetzt, wenn vom verwendeten Browser unterstützt.

Wie vor beschrieben stehen auch entsprechende Apps für mobile Geräte zur Verfügung. Bei den mobilen Geräten ist darauf zu achten, dass die Pixelgrößen des Zielsystems exakt eingehalten werden. Andernfalls läßt sich die GUI nicht darstellen.

Werden mehrere .gui Files in der Ordnerstruktur abgelegt, können diese mittels Auswahl im Webbrowser abgerufen werden.



Der Webserver ist durch aufruf der IP Adresse des entsprechenden nomos systems erreichbar.

5.3 Timer Support

Das nomos system unterstützt Timer und Kalender Funktionen. Timer dienen z.B. der zyklischen Abfrage bestimmter Informationen. Kalender führen zeitlich definierte Abläufe aus (Wochenuhr). Timer wie Kalender können im laufenden Prozess definiert, verändert, gestartet oder gestoppt werden.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	timer.csv
Länge:	256 Timer möglich
Event:	LOG
Export:	nein
Lizenz:	nein

Die `timer.csv` **besteht aus 2 Sektionen:**

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[TIMERS]; Definition der Timer/Kalender Einträge

Aufbau:

- **[CONFIG]; Sektion**

  ACTIVE;YES „`YES`" = aktiv (default),
  „`NO`" = nicht aktiv

  DEBUG;NO „`YES`" erzeugt einen Logeintrag, wenn ein Timer feuert.
  „`NO`", Es werden nur Start bzw. Stopp eines Timers geloggt (default).

- **[TIMERS]; Sektion** {Name};{Typ};{time/s} o.
  {{Datums-String}~{{Varianz}};{Action};{Optionen}

Erläuterungen :

{Name} Bezeichnung des Timers

{Typ} Typ des Timers, mögliche Werte:

INTERVAL zyklischer Timer feuert im Abstand der `{time/s}`

IMMEDIATE wie vor, feuert jedoch erstmalig direkt bei Start

ONESHOT feuert einmalig nach `{time/s}`

CALENDAR Wochenschaltuhr, Der Typ kann mit drei Buchstaben abgekürzt werden, `CAL` für `CALENDAR` als Beispiel.

{time/s} oder Intervall in Sekunden (max. `86400s` (= 1Tag) möglich) wird verwendet bei folgendem `{Typ}`:
`INTERVAL, IMMEDIATE, ONESHOT`.

oder

{Datums-String}~{Varianz} Datums-String bei Verwendung des `{Typ} = CALENDAR`
Beispiele für den Datums-String werden nachfolgend erläutert.

{Action} auszuführende Kommando Sequenz ..

{Optionen} weitere Optionen zur Timerkonfiguration, z.Zt. möglich:

AUTOSTART startet den Timer beim Start des nomos systems automatisch.

Beispiel für die Defintionen (einfacher Timer):

[TIMERS];IVT;INTERVAL;1;<ITUNES><GETPTIME><GETPTIME{%02m:%02s}><GETCTIME><GETCTIME{%02m:%02s}><GETREPEAT><GETSHUFFLE></ITUNES>;AUTOSTART

Startet den Timer `IVT` automatisch bei Systemstart und führt die Itunes-Befehle im Ein-Sekunden-Takt aus.

CLEARSCREEN;ONESHOT;5;<WINDOW><SELECT=3><RELEASE></WINDOW>; Reserviert den Timer `CLEARSCREEN`. Nach dem Befehl `<TIMER><START= CLEARSCREEN></TIMER>` einmalig und führt nach fünf Sekunden entsprechende Befehlsfolge aus.

Beispiel {Datums-String} - mögliche Inhalte (CALENDAR):

{nichts} feuert jeden Tag um 0 Uhr

{13} feuert jeden Tag um 13 Uhr

{13:04} feuert jeden Tag um 13 Uhr und 4 Minuten

{13:04:05} feuert jeden Tag um 13 Uhr, 4 Minuten und 5 Sekunden

{MONDAY} feuert jeden Montag um 0 Uhr

{WEDNESDAY,13:04} feuert jeden Mittwoch um 13 Uhr und 4 Minuten

{WEDNESDAY,MONDAY,13} feuert jeden Mittwoch und Montag um 13 Uhr

{THURSDAY,13,29.11.} feuert jeden Donnerstag um 13 Uhr -> Wenn Wochentag und Datum angegeben werden, geht der Wochentag vor.

Beispiel {Datums-String}~{Varianz} - Random-Calendar-Timer, mögliche Inhalte (CALENDAR):

Die bisherigen `CALENDAR`-Timer lassen sich um eine beliebig zu wählende Unschärfe (=Varianz) erweitern. Dabei wird für jedes Timer-Feuern eine neue zufällige Zeit erzeugt, er ist damit nicht voraussagbar. Dies ist z.B. für Anwesenheitssimulationen sinnvoll. Die Varianz wird im Timer-String direkt an die Zeit angefügt, damit ist die Angabe einer Zeit in diesem Fall zwingend.

Die fix definierte Zeit beschreibt dann die Ziel-Zeit, die Varianz die Breite der Abweichung von der Ziel-Zeit

Syntaxvarianten: {Datums-String}~{Varianz} ~{Sekunden}
~{Minuten}:{Sekunden}
~{Stunden}:{Minuten}:{Sekunden}

01:00~30:00,SAT,SUN Die Ziel-Zeit ist 01:00 Uhr am Wochenende (Samstag und Sonntag), diese wird um eine Varianz von 30 Minuten erweitert. Der Timer feuert also irgendwann zwischen 0:45 Uhr und 01:15 Uhr.

SUN,MON,TUE,WED,THU,23:00~900 Die Ziel-Zeit ist 23:00 Uhr an 5 verschiedenen Wochentagen, diese wird um eine Varianz von 900 Sekunden (15 Minuten) erweitert. Der Timer feuert also irgendwann zwischen 22:52:30 und 23:07:30 Uhr.

SAT,SUN,22:00~01:00:00 Die Ziel-Zeit ist 22:00 Uhr am Wochenende (Samstag und Sonntag), diese wird um eine Varianz von 1 Stunde erweitert. Der Timer feuert also irgendwann zwischen 21:30 Uhr und 22:30 Uhr.

Die Reihenfolge von Uhrzeit, Wochentag und Datum im `{Datums-String}` ist egal. Die Uhrzeit wird immer als alleinstehende Zahl (mit oder ohne „:") erkannt, das Datum immer an mindestens einem ".".

{13,12.} feuert jeden 12. des Monats um 13 Uhr

{24.12.,18} feuert jeden Heiligabend um 18:00 Uhr

{13,25.12} feuert am 1. Weihnachtsfeiertag zum Mittagessen

Die Wochentage können auch mit den ersten drei Buchstaben (zwingend) abgekürzt werden: `MON,TUE,WED` usw. Ein `CALENDAR` -Timer muss, sofern die `AUTOSTART` Option nicht gesetzt ist, wie alle anderen Timer, mit `START={Timername}` gestartet werden.

Beispiel (CALENDAR Timer):

WAKEUP;CALENDAR;MON,TUE,WED,THU,FRI,07:00;<ITUNES> <PLAYPLAYLIST=EarlyMorningMusik></ITUNES>;AUTOSTART

Startet den Timer `WAKEUP` automatisch bei Systemstart und führt den Itunes-Befehl an jedem Werktag um 07:00 Uhr aus.

BACHLAUF;CALENDAR;13:00;<KNX><SETVALUE=1/2/23,1></KNX>;AUTOSTART
Startet den Timer `WAKEUP` automatisch bei Systemstart und führt den Befehl an jedem Tag um 13:00 Uhr aus.

Heizung WZ Nacht;CALENDAR;SAT,SUN,22:00~01:00:00;<SYS><LOCKVARS=OFF> <SETVAR=WZTEMP,17.00><LOCKVARS=ON></SYS>;AUTOSTART;

Die Zielzeit ist 22 Uhr, die Varianz beträgt 1 Stunde: Der Timer feuert irgendwann zwischen 21:30 und 22:30.

Ausgaben im LOG:

Die Aktivitäten der Timer können im Log verfolgt werden. Erweiterte Ausgaben werden im Log ausgegeben, sofern `DEBUG;YES` gesetzt ist.

Info über einen Timer der gestartet wurde.

```
18:48:33.727 - timer: calendar timer WAKEUP started
```

Info, wann der Timer das nächste mal ausgeführt wird.

```
18:48:33.727 - timer WAKEUP will fire at Saturday, 04.01.2014, 18:50:00
```

Info über die Ausführung des Timers (DEBUG;YES)

```
18:50:00.086 - timer: executing WAKEUP: <SCRIPT><RUN=itunes.myh></SCRIPT>
```

Timer können dynamisch während der Laufzeit manipuliert werden. Zu diesem Zweck dient folgender Befehlssatz.

Es existiert eine Befehlsklasse TIMER mit folgenden Befehlen:

Kommando	Argument	Beschreibung
CREATE#	{Name},{Timertyp}, {Wert oder Datums- String}, {Timeraction}	erzeugt einen neuen Timer und schreibt ihn in die `timer.csv`
DELETE#	{Name}	loescht einen existierenden Timer (und stoppt ihn ggfs.)
DISABLE#	{Name}	entfernt `AUTOSTART` aus der `timer.csv` und stoppt den Timer, falls er laufen sollte
ENABLE#	{Name}	schreibt `AUTOSTART` hinter einen existierenden Timer in der `timer.csv` und startet ihn, falls noch nicht geschehen
LIST		listet alle aktiven Timer
LISTALL		listet alle definierten Timer mit ihren Namen auf
RESET#	{Name}	setzt den Timer `{Name}` zurueck
START#	{Name}{Name}	startet den Timer `{Name}`
STOP#	{Name}{Name}	stoppt den Timer `{Name}`
LIST#	{Name}	listet alle aktiven Timer `{Name}`

Namen für Timer `{TimerName}` können beliebig verwendet werden. **Beispiele**

<TIMER><LIST></TIMER> Generiert eine Liste aller aktiven Timer. Wenn kein Timer aktiv, bleibt der Listeneintrag leer.

<TIMER><CREATE=WAKEUP,CAL,07:00,<SYS><SAY=Good Morning></SYS>></TIMER> erzeugt einen CALENDAR-Timer, der jeden Morgen um 07:00 Uhr "Guten Morgen" sagt. Timertyp kann ebenfalls mit den ersten drei Buchstaben abgekürzt werden. Mit `CREATE` angelegte Timer-Actions werden nicht mehr verändert.

<TIMER><ENABLE=WAKEUP></TIMER> Aktiviert vorherig erstellten Timer und setzt ihn auf `AUTOSTART`

Korrekte Timer Definitionen werden durch folgendes Reply bestätigt:

<TIMER>CREATE=|OK</TIMER>

Vorhandene Timer können nicht verändert werden. Um einen Timer zu ändern, muss er mittels dem `DELETE=` Befehl gelöscht werden. Wird der `CREATE=` Befehl auf einen bereits vorhandenen Timer Namen ausgeführt, erfolgt eine Fehlermeldung:

```
22:05:10.303 - timer: ERROR: timer UP already exists
```

5.4 Counter Support

Das nomos system unterstützt auch Counter. Counter werden wie Sysvars behandelt, werden jedoch nie gespeichert. Mit jedem Auslesen eines Counters wird der vorherige Wert um eine einstellbare Schrittweite verändert.

Für die Manipulation der Counter, stehen folgende `<SYS>` Befehle zur Verfügung:

Kommando	Argument	Beschreibung
SETCOUNTER=	{Name}{,Startwert} {,Schrittweite}	initialisiert den Counter `{Name}.{Startwert}` und `{Schrittweite}` sind optional. Default ist 0 für `{Startwert}` und 1 für `{Schrittweite}`
SETCOUNTERSTEP=	{Name} {,Schrittweite}	ändert die Schrittweite eines bestehenden Counters `{Name}`, 0 für `{,Schrittweite}` hält den Counter an.
GETCOUNTER=	{Name}	liefert den aktuellen Wert des Counters {Name}, ohne sie zu verändern
GETALLCOUNTERS		liefert die aktuellen Werte aller Counter, ohne diese zu verändern

In Befehlssequenzen kann mit `[{Name}]` auf den Counterwert zurückgegriffen werden (exakt wie bei den `Sysvars`, jedoch wird hier im Anschluß daran der jeweilige Counterwert um die Schrittweite verändert.)

Beispiele:

<SYS><SETCOUNTER=COUNT,0,10></SYS> Erstellt den Counter mit dem Namen `[COUNT]`. Der Counter beginnt bei Null und hat eine Schrittweite von 10.

<SYS><GETCOUNTER=COUNT></SYS> Liefert den aktuellen Wert des Counters mit dem Namen `[COUNT]`.

Beispiel für die Verwendung eines Counters um iTunes Titel als Laufschrift an den KNX zu übergeben.

In die `init.myh` den folgenden Befehl eintragen:

<SYS><SETCOUNTER=ITC></SYS>* Dies bewirkt, dass der Counter mit dem Namen ITC bei Systemstart initialisiert wird.

In die `timer.csv` kommt:

ITT;INTERVAL;1;<ITUNES><GETTITLE{%c[ITC],14}></ITUNES>;AUTOSTART Dieser Eintrag definiert einen Timer „`ITT`" der in einem Intervall von einer Sekunde eine iTunes `GETTITLE` Abfrage auslöst, deren Rückgabewert durch den Parameter `%c` manipuliert wird. Der Parameter `%c` benötigt einen Index und eine Länge. Der Index wird von dem voherig definierten Counter (`[ITC]`) beschrieben und die Läge mit dem fixen Wert `14` belegt. Durch die Autostartfunktion wird der Counter automatisch bei Systemstart aktiviert.

Nun muss noch folgende EventServer Definition eingerichtet werden: `.\events\lauftext.csv:`

```
[CONFIG];;
TRIGGERIP;INTERNAL;
```

```
TRIGGERPORT;OUT
TRIGGERMODE;ASCII;
MATCHING;FULL;
```

```
[TRIGGERS];;
GETTITLE{\%}=\*|;<KNX><SETVALUE=1/5/5,\#></KNX>
```

Der `TRIGGER` parst auf `GETTITLE`, also den Namen des aktuell angespielten iTunes Titels. Der Wert der gematcht wurde, wird an die KNX-Adresse `1/5/5` geschrieben. In der `.esf` Typdefinition muss diese Adresse natürlich als ‘Character String’ (14 Byte)EIS 15 definiert sein.

Aufgrund der Abfrage, die über den Timer sekündlich initiiert wird, wird der Counter `[ITC]` immer um seine Schrittweite erhöht. Da bei der Initialisierung des Counters keine Schrittweite angegeben wurde ist der Defaultwert, also 1, angenommen worden. Durch den Parameter `%c` ist der Ausgabestring manipuliert worden. Es werden nur 14 Zeichen ab einem bestimmten Index ausgegeben. Der Index wird von dem Counter beschrieben, der bei jeder (sekündlichen) Abfrage um 1 erhöht wird. Die besonderen Eigenschaften des Parameters `%c` verhindern ein Überlauf. Siehe hierzu die Beschreibung zur Verwendung von Parametern in den Formatierungsoptionen.

5.5 Logik

Das nomos system verfügt über eine integrierte, leistungsstarke Logik Engine. Die Logik Engine unterstützt im Allgemeinen boolsche Ausdrücke wie zB `NOT, AND, OR` sowie vergleichende Logik wie `GLEICH`, `UNGLEICH`, `GROESSER`, `KLEINER`, `KLEINER-GLEICH`, `GROESSER-GLEICH`. Die Funktionen können beliebig angewendet werden. Daraus ergibt sich eine Vielzahl von möglichen Anwendungen. Eine Besonderheit der Logik Engine ist die typenlose Behandlung der Operanden. Dies macht es möglich, sämtliche Informationen aus den verschiedenen Protokollen über den `IN/OUT/BC` Kanal zu filtern und direkt in einer logischen Funktion zu verarbeiten. Ebenso die Logik Engine auf die Joins der `mremote.csv`. Somit kann ein Digital-, Analog- oder Serialjoin direkt als `OBJEKT` verarbeitet werden. Gleiches gilt für Systemvariablen.

Aufbau der Definitionstabelle:

Typ:	csv Datei
Ort:	.\{projectPath}\misc\
Name:	logic.csv
Länge:	1000 Zeilen
Event/Trigger:	IN/OUT/BC - mremote - sysvars
Export:	-
Lizenz:	-

Die `logic.csv` besteht aus 3 Sektionen:

[CONFIG]; allgemeine Konfiguration der Interface Parameter

[OBJECTS]; Definition der Objekte, die als Operand für die logische Abarbeitung benötigt werden.

[LOGIC]; Definition der logischen Ausdrücke.

Aufbau:

- **[CONFIG];Sektion**

  ACTIVE;YES Aktiviert/Deaktiviert die Logik Engine
  „`YES`" (Default)= aktiv,
  „`NO`" = nicht aktiv

  DEBUG;YES Erweiterte LOG Ausgabe
  „`YES`" = aktiv,
  „`NO`" (Default) = nicht aktiv

- **[OBJECTS];Sektion**

  {Objektname};{Match};{Startwert};{Kanal};ONCHANGE; Hier werden Objekte definiert, die für die Logik Engine validiert werden sollen. Werden JOINS oder Systemvariablen benutzt, müssen diese hier nicht weiter berücksichtigt werden

  Erläuterungen:

  {Objektname} Name des Objektes

  {Match} Match-String zu Erfassung des Wertes für das Objekt

  {Startwert} Vorbelegung des Objektwertes (optional, default: `0`)

  {Kanal} Parser-Kanäle (`IN/OUT/BC`), auf denen gematcht werden soll (optional, default: `IN,OUT,BC`)

ONCHANGE triggert die Funktion (siehe unten) nur bei Werteänderungen (optional)

- **[LOGIC];Sektion** {logischer Ausdruck};{positive Aktion};{negative Aktion};{Triggerliste};ONCHANGE;

  Erläuterungen:

  {logischer Ausdruck} Verknuepfungen von Objekten, Sysvars und Konstanten

  {positive Aktion} Aktion, die bei einem logischen "wahr" des `{logischen Ausdrucks}` ausgefuehrt werden soll, Objekte koennen mit [[ ]] eingefuegt werden.

  {negative Aktion} Aktion, die bei einem logischen "falsch" des `{logischen Ausdrucks}` ausgefuehrt werden soll, Objekte können mit [[ ]] eingefuegt werden.

  {Triggerliste} Liste aller `Objekte` und `Sysvars`, die die Evaluierung des `{logischen Ausdrucks}` anstossen (optional, default:alle)

  ONCHANGE Triggert die `positive` bzw. `negative` Action nur dann, wenn sich der Wert des `logischen Ausdrucks` geaendert hat (wahr -> falsch oder falsch -> wahr), optional.

  Operanden von {logischer Ausdruck}:

  {Objektname} Inhalt des Objektes `{Objektname}`

  {{D/S/A...}} Inhalt eines Join aus der `mremote.csv`. `{D512}` für Digitaljoin 512 als Beispiel.

  [{Sysvarname}] Inhalt der Sysvar `{Sysvarname}`

  "{Konstante}" konstanter Wert

  logische Verknuepfungen:

  ! `"NOT"`, logische Negation, kann vor Operanden und Klammern stehen

  | oder || `"ODER"`, Operand a oder Operand b sind wahr

  & oder && `"UND"`, Operand a und Operand b sind wahr

  Vergleicher:

  = oder == `"GLEICH"`, Operand a ist gleich Operand b

  != oder <> `"UNGLEICH"`, Operand a ist ungleich Operand b

  > oder >> `"GROESSER"`, Operand a ist grösser Operand b (auf Zahlenwerte bezogen)

  < oder << `"KLEINER"`, Operand a ist kleiner Operand b (auf Zahlenwerte bezogen)

  >= oder => `"GROESSER-GLEICH"`, Operand a ist grösser oder gleich Operand b (auf Zahlenwerte bezogen)

  <= oder =< `"KLEINER-GLEICH"`, Operand a ist kleiner oder gleich Operand b (auf Zahlenwerte bezogen)

  Rechenvorschrift:

() die Evaluierung von gesetzten Klammern hat Vorrang
Fliesskommazahlen werden ebenfalls unterstützt

Beispiele für die Definitionen [OBJECT] Sektion: {Objektname};{Match};{Startwert};{Kanal};ONCHANGE;

MYVOL;<SYS>\%|VOL=*|;;BC,OUT;ONCHANGE; Definiert das Objekt `MYVOL` welches auf den Inhalt der Systemlautstaerke aus dem OUT- und/oder BC-Kanal matcht.

DP1;<KNX><1/1/6\%=*></KNX>;;BC; Definiert das Objekt `DP1` welches auf den Inhalt der KNX Gruppenadresse `1/1/6` aus dem BC-Kanal matcht.

Beispiele für die Definitionen [LOGIC] Sektion: {logischer Ausdruck};{positive Aktion};{negative Aktion};{Triggerliste};ONCHANGE;

(MYVOL > "50") & (MYVOL > [WANTEDVOLUME]) & ![DONTSAY];<SYS><SAY= Your Volume is [[MYVOL]], which is higher than[WANTEDVOLUME]><SYS>;;MYVOL;ONCHANGE;

Führt die Action einmal aus, wenn das Objekt `MYVOL` seinen Wert ändert und sowohl ueber dem Wert `50` als auch ueber dem Inhalt der Sysvar `WANTEDVOLUME` liegt und die Sysvar `DONTSAY` logisch falsch ist. Änderungen bei `[WANTEDVOLUME]` oder `[DONTSAY]` triggern die Evaluierung nicht, weil MYVOL explizit als Trigger definiert wurde

{D351} || {D352} || {D301} || D313});<SYS><SETDIGITALJOIN=353,1><SYS>;<SYS> <SETDIGITALJOIN=353,0></SYS>;;ONCHANGE;

Führt die positive Aktion aus, wenn einer der `Digitaljoins 351, 352, 301, 313` den Wert „`1`" annimmt `(ODER)`. Wenn alle Digitaljoins den Wert „`0`" annehmen, wird die negative Aktion ausgeführt. Die Aktionen werden nur bei einer Änderung von „`0`" nach „`1`" oder „`1`" nach „`0`" ausgeführt, da die `ONCHANGE` Option gesetzt ist.

Sonstige Hinweise:

Ein `OBJEKT` bzw. eine `SYSVAR` mit folgenden Inhalten ist logisch "wahr": `{Wert > 0}, TRUE,ON,YES`.

Ein `OBJEKT` bzw. eine `SYSVAR` mit folgenden Inhalten ist logisch "falsch": `0,FALSE,OFF,NO`.

Zum Zeitpunkt der Evaluierung nicht definierte `SYSVARS` werden mit dem Wert `"0"` angenommen.

In der Triggerliste dürfen beliebige `OBJEKTE` und `SYSVARS` stehen, auch wenn sie nicht im logischen Ausdruck vorkommen

Die Auswertung der Joins erfolgt "direkt", also ohne Betrachtung der `mremote.csv`, damit finden z.B. evtl. definierte Skalierungen bei aJOINS keine Anwendung. Es wird nur auf Werte gematcht, die vom Daemon an die GUI gesendet werden. Nur so lässt sich z.B. eine Unterscheidung von `TOGGLE`-Buttons umsetzen. Ausserdem lässt sich so indirekt mit `AUTOFEEDBACK` und `DELAYFEEDBACK` beeinflussen, wann die Logik getriggert wird.

Die übrige Behandlung der Joins (Trigger, Bedingungen usw.) ist analog zu `OBJEKTEN und SYSVARS`.

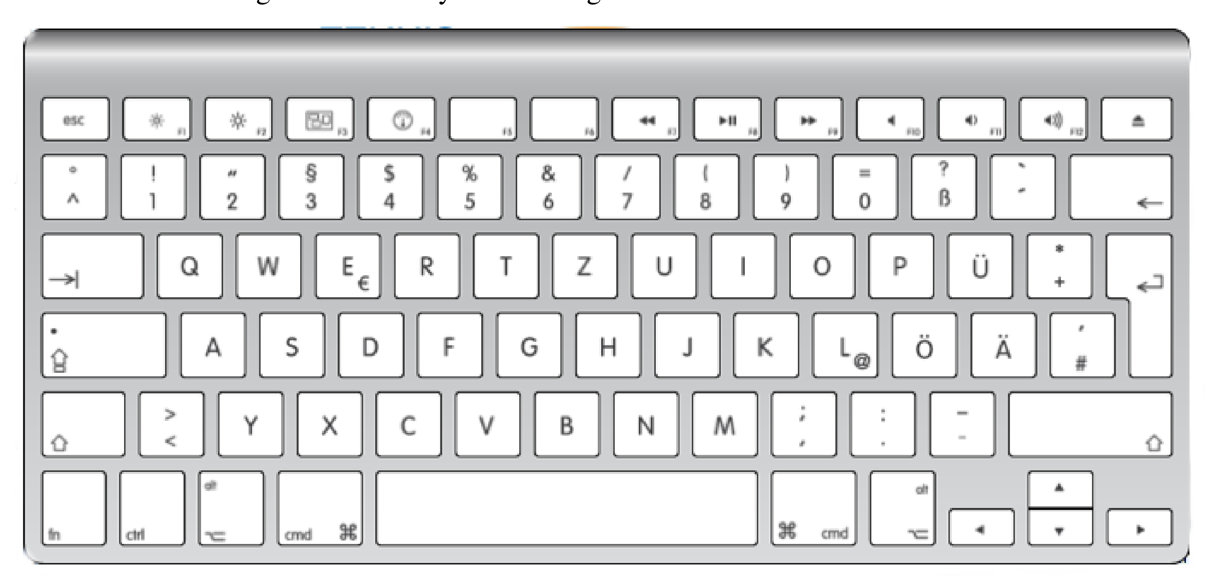
5.6 OSD Keyboard

Das nomos system (Apple OS X Version only) verfügt über ein virtuelles Keyboard. Dieses OSD-Keyboard ist Bestandteil der nomos GUI. Das OSD Keyboard bietet sich hervorragend für Systeme mit Touchscreen.

Die Breite des Keyboards beträgt immer 50% der aktuellen Bildschirmbreite, die Proportionen des Keyboards bleiben unabhängig vom Bildseitenverhältnis immer konstant. Befindet sich der Mauszeiger ausserhalb des Keyboards, wird es zu 70% transparent.

Das Keyboard lässt sich überall auf dem Hintergrund anklicken und über den Bildschirm ziehen.Das Keyboard ist immer das oberste Fenster.

Es bekommt immer die Applikation die Key-Events gesendet, die entweder beim Aufruf des Keyboards oder als der Mauszeiger über das Keyboard bewegt wurde den Focus hatte.



Doppelklick auf das oberste Zehntel (über den F-Tasten) des Keyboards lässt das Keyboard verschwinden Einschränkung: die Fn-Taste funktioniert noch nicht.

Für die Steuerung des OSD Keyboard, stehen folgende <SYS> Befehle zur Verfügung:

SHOWKEYBOARD	Blendet die Eingabehilfe ein (Keyboard)
HIDEKEYBOARD	Blendet die Eingabehilfe aus (Keyboard)

Beispiele:

<SYS><SHOWKEYBOARD></SYS> Öffnet die virtuelle Tastatur auf einem Apple OSX System

<SYS><HIDEKEYBOARD></SYS> Blendet die virtuelle Tastatur auf einem Apple OSX System wieder aus.

5.7 Systemerkennung im Netzwerk - ZeroConf

Das nomos system verfügt über eine automatische Systemerkennung im Netzwerk (Zero configuration networking - Zeroconf). Es erkennen sich alle nomos und verwandte OEM Versionen selbständig und gegenseitig.

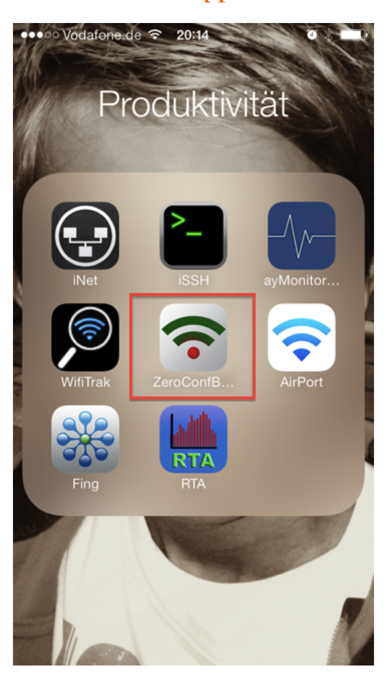
Eine Liste der IPs aller gefundenen Systeme kann man mit folgendem Befehl abfragen:

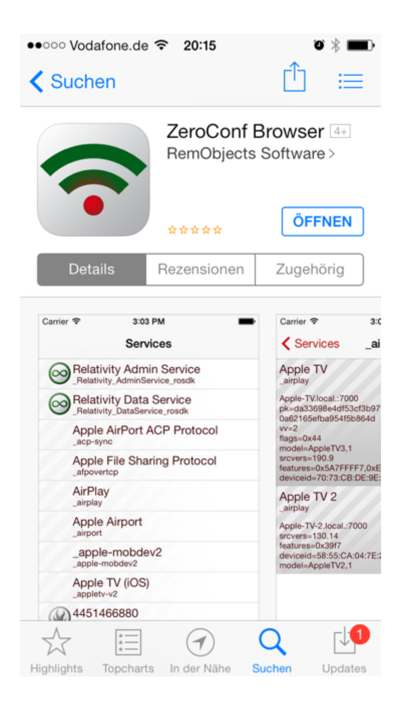
<SYS><GETSYSTEMLIST></SYS> Abfrage der verfügbaren nomos systeme im Netzwerk

Mögliche Antwort: <SYS>GETSYSTEMLIST=192.168.50.60 | 192.168.50.61 | 192.168.50.55 | OK</SYS>

Ausserdem kann man sich auf sein iPhone/iPad die kostenlose App ROZeroConf herunterladen. Das nomos System stellt für die App zwei Dienste zur Verfügung. `_rmtsysctrl` listet alle laufenden Daemon (inkl. OEM-Name, Seriennummer) sowie exportierte Commandserver im Netzwerk auf. `_rmtsysinfo` hingegen liefert MAC-Adresse, IP, Seriennummer, Name und Version der Box auch dann, wenn keine gültige Lizenz vorhanden ist.

Für die erstmalige Inbetriebnahme der nomos Boxen wird ein DHCP-Server im Netzwerk benötigt. Die vergebene IP erfährt man entweder über das Log des DHCP-Servers oder über das ZeroConf-Protokoll. Die ZeroConf App ist hierbei für die Erstinbetriebnahme der nomos Boxen äußerst hilfreich.





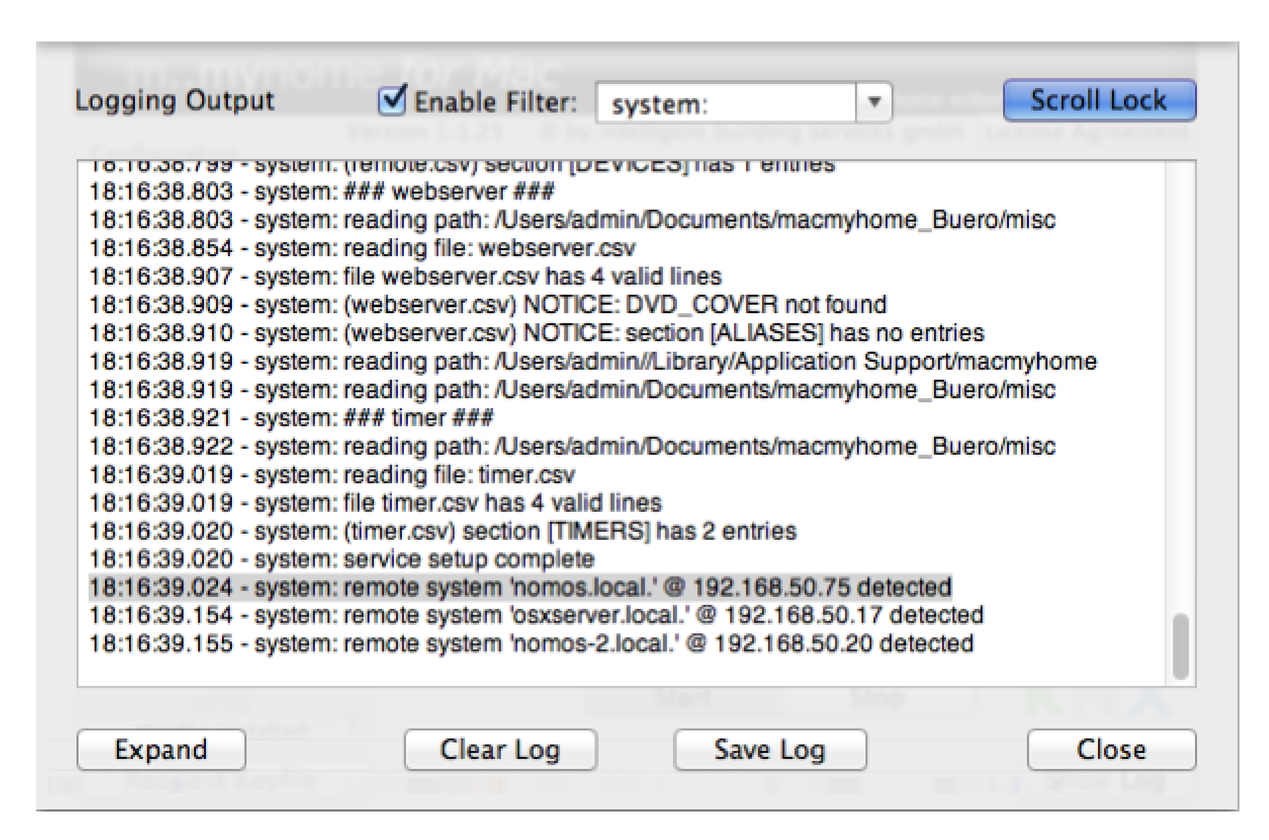


Abbildungen des ZeroConf Browsers

Gefundene Systeme erscheinen mit Name und IP-Adresse auch im LOG. Es erscheint im LOG zB folgende Meldung. Die Meldung wird automatisch nach jedem Start eines Systems automatisch an alle vorhandenen Systeme gesendet:

```
18:16:39.224 - system: remote system 'nomos.local.' @ 192.168.50.75 detected
18:16:39.395 - system: remote system 'nomosBox.local.' @ 192.168.50.61 detected
18:16:39.524 - system: remote system 'iMac-Mike.local.' @ 192.168.50.30 detected
18:16:39.922 - system: remote system 'MacBookAir.local.' @ 192.168.50.128 detected
18:16:39.924 - system: remote system 'Mikes-MacBook-2.local.' @ 192.168.50.124 detected
```

Es empfiehlt sich, die LOG Ausgabe nach einem Systemstart (Initial LOG) anzeigen zu lassen.



Alle verfügbaren Systeme werden unter `system: remote system` aufgelistet.

5.8 VPN Zugriff

Das nomos system verfügt über eine integrierte Support Schnittstelle (Open VPN). Der Nutzer kann per Befehl oder per Dialog (Wizard Versionen) den Zugang freigeben. Ist der Zugang freigegeben, ist das nomos Support Team in der Lage die Box über einen VPN Kanal zu erreichen. Hierbei ist sichergestellt, dass in Ihrer Netzwerkstruktur ausschließlich nur das nomos system erreichbar ist /1er Netzwerk.

Auf den nomos Box Versionen ist die VPN Funktion vorinstalliert. Für die Apple OS X Version ist ein entsprechendes VPN-Package verfügbar. Das VPN-Package für Apple OS X basiert auf der Tunnelblick-Applikation (http://www.tunnelblick.net).

Die Applikation ist entsprechend vorkonfiguriert und für die nomos Anwendung modifiziert. Eine aktive VPN- Verbindung erkennt man oben rechts in der Systemleiste an einem Tunnelsymbol, in dem grüne Lichtstrahlen zu sehen sind. Rechtsklick und "Tunnelblick beenden" trennt die VPN-Verbindung, ein Doppelklick auf die Tunnelblick-Applikation im Programme-Ordner startet sie.

Um die Steuerung durch das nomos system zu gewährleisten, darf die Applikation nicht umbenannt oder verschoben werden! Ebenso sollten keine manuellen Updates der Applikation gemacht werden, da eine spezielle Version der Applikation benötigt wird.

Die Steuerung erfolgt wie auf den nomos Box Versionen mit den Befehlen:

<SYS><VPN=ON></SYS> Initiiert die VPN Verbindung

<SYS><VPN=OFF></SYS> Beendet die VPN Verbindung

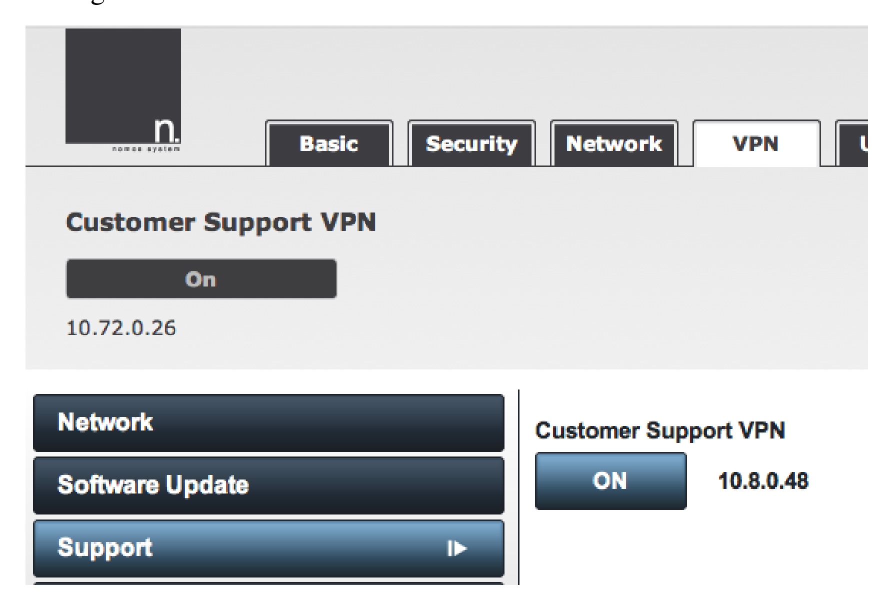
Bitte beachten Sie, dass eine aktive VPN Verbindung auch nach einem Restart des Systems verfügbar bleibt. Wollen Sie den VPN Zugang dauerhaft unterbinden, müssen sie den Dienst ausschalten.

Ist die VPN Verbindung geöffnet, kann die zugeteilte VPN IP per Befehl abgefragt werden. Die VPN IP Adresse wird fest zugeteilt. Vorerst ist es so, dass die VPN IP Adresse an die Seriennummer des nomos Systems gebunden ist. Somit bleibt die zugeteilte IP Adresse längerfristig bestehen.

<SYS><GETVPNIP></SYS> liefert die VPN-IP, falls eine VPN-Verbindung besteht

mögliche Antwortsequenz reply sequence: <SYS>GETVPNIP=10.72.0.26|OK</SYS>

Die VPN IP Adresse müssen Sie dem Supportteam mitteilen. Bei den nomos Box Versionen erhalten Sie diese Information ebenfalls über die Basis Configuration in dem Bereich VPN/Support. Je nach Version können die Abbildungen unterschiedlich ausfallen.



Zur Zeit verfügt ausschließlich das nomos Supportteam über die Möglichkeit des VPN Zuganges. In Zukunft wird es ein Portal geben, wo der Anwender sein System registrieren und ebenfalls auf diese Schnittstelle zugreifen kann.

KAPITEL 6

TABELLE key codes SYSTEM EVENTS

APPLESCRIPT Deutsche Apple USB-Tastatur, evtl. auch für andere Tastaturen gültig (alle Angaben ohne Gewähr):

key code 1 = "s"!	key code 75 = NUM [/]
key code 2 = "d"	key code 76 = NUM [Enter]
key code 3 = "f"	key code 77 = [ArrowUp]
key code 4 = "h"	key code 78 = NUM [-]
key code 5 = "g"	key code 79
key code 6 = "y"	key code 80
key code 7 = "x"	key code 81 = NUM [=]
key code 8 = "c"	key code 82 = NUM [0]
key code 9 = "v"	key code 83 = NUM [1]
key code 10 = "^"	key code 84 = NUM [2]
key code 11 = "b"	key code 85 = NUM [3]
key code 12 = "q"	key code 86 = NUM [4]
key code 13 = "w"	key code 87 = NUM [5]
key code 14 = "e"	key code 88 = NUM [6]
key code 15 = "r"	key code 89 = NUM [7]
key code 16 = "z"	key code 90
key code 17 = "t"	key code 91 = NUM [8]
key code 18 = "1"	key code 92 = NUM [9]
key code 19 = "2"	key code 93
key code 20 = "3"	key code 94
key code 21 = "4"	key code 95
key code 22 = "6"	key code 96 = [F 5]
key code 23 = "5"	key code 97 = [F 6]
key code 24 = " "	key code 98 = [F 7]
key code 25 = "9"	key code 99 = [F 3]
key code 26 = "7"	key code 101 = [F 9]
key code 27 = "ß"	key code 100 = [F 8]
key code 28 = "8"	key code 102
key code 29 = "0"	key code 103 = [F11]
key code 30 = "+"	key code 104
key code 31 = "o"	key code 105 = [F13]
key code 32 = "u"	key code 106 = [F16]
key code 33 = "ü"	key code 107 = [F14]
key code 34 = "i"	key code 108

key code 35 = “p”	key code 109 = [F10]
key code 36 = [Return]	key code 110
key code 37 = “l”	key code 111 = [F12]
key code 38 = “j”	key code 112
key code 39 = “ä”	key code 113 = [F15]
key code 40 = “k”	key code 114 = [Help]
key code 41 = “ö”	key code 115 = [PgUp]
key code 42 = “#”	key code 116 = [Home]
key code 43 = ”,”	key code 117 = [Fwd-Del]
key code 44 = “-“	key code 118 = [F 4]
key code 45 = “n”	key code 119 = [PgDwn]
key code 46 = “m”	key code 120 = [F 2]
key code 47 = ”.”	key code 121 = [End]
key code 48 = [Tab]	key code 122 = [F 1]
key code 49 = [Space]	key code 123 = [ArrowLeft]
key code 50 = “<”	key code 124 = [ArrowRight]
key code 51 = [Bkw-Del]	key code 125 = [ArrowDown]
key code 52 = [Return]	key code 126 = [ArrowUp]
key code 55 = [Command]	key code 127
key code 54 = [Command]	key code 128 = “a”
key code 53 = [Escape] esc	
key code 54 = [Command]	(ab 129 beginnt es wieder von vorn: 129 = “s”, 130 = d”, usw.
key code 55 = [Command]	
key code 56 = [Shift]	
key code 57 = [ShiftLock]	
key code 58 = [Alt]	
key code 59 = [Ctrl] ctrl	
key code 60 = [Shift]	
key code 61 = [Alt]	
key code 62 = [Ctrl] ctrl	
key code 64	
key code 65 = NUM [,]key code 66 = [ArrowRight]	
key code 67 = NUM [*]	
key code 68	
key code 69 = NUM [+]	
key code 70 = [ArrowLeft]	
key code 71 = NUM [X]	
key code 72 = [ArrowDown]	
key code 73	
key code 74	

KAPITEL 7

Workarounds, Previews

Unter Workarounds verstehen wir Möglichkeiten, die sich in der Erprobung befinden. Wir möchten Ihnen diese Funktionen nicht vorenthalten und bieten Ihnen an, diese zu testen und bitten ggf. um Ihr Feedback. Bitte beachten Sie, dass kein Anspruch auf Evaluierung dieser Funktionen besteht. Es kann also sein, dass wir nachfolgend beschriebene Möglichkeiten nicht weiter verfolgen.

7.1 DVD-Copy-Window

Die Copy-Funktion erwartet im `gui` -Ordner eine Datei `DVDCopy.csv` mit folgendem Inhalt:

[CONFIG];

ACTIVE;YES aktiviert die Kopierfunktion (default: `NO`)

PATH;{Pfad zum DVD Sammelordner} Der Pfad muss bereits vorhanden sein, beachten Sie die korrekte Namensgebung der Ordner. Der im „Finder“ sichtbare Ordner „`Filme`“ heißt real „`Movies`“ . Ein Beispiel für den lokalen Pfad der Standard-Ordnerstruktur wäre z.B.: `/Users/mmh/Movies`

Funktionsweise:

Legt man eine Disk ein, wird zunächst geprüft, ob sich auf der Disk ein `VIDEO_TS` -Ordner befindet. Falls ja, wird geprüft ob in PATH bereits ein Ordner mit gleichem Namen existiert. Ist dies nicht der Fall, öffnet sich oben-mittig ein kleines Fenster mit dem DVD-Namen. Ignoriert man dieses Fenster, schließt es sich nach 10 Sekunden automatisch. Klickt man auf “Copy”, beginnt der Kopiervorgang und der Fortschrittsbalken wächst, anschließend wird die DVD ausgeworfen.

Klickt man (egal zu welchem Zeitpunkt) auf “Abort”, wird der Kopiervorgang abgebrochen, der unvollständige, bereits erstellte Zielordner wieder gelöscht und die DVD ausgeworfen.

Die Funktion greift auf die Apple OSX interne Kopierfunktion zurück. Zukünftig kann auch ein Pfad zu einem Shellscript angegeben werden. Hierüber können dann auch andere Kopierprogramme aufgerufen und evtl. notwendige Parameter übergeben werden.

Kopierschutz: Mit dieser Funktion können keine kopiergeschützten Medien kopiert werden!

7.2 iTunes Remote

Apple hat mit der neuen iPod Touch und iPhone Geräte-Generation das „Tor“ für native Softwareanwendungen Dritter geöffnet. Apple selbst hat die Software „Remote“ (kostenlos zu laden über den iTunes Applikation Store) zur Verfügung gestellt. Apple Remote ermöglicht das komfortable Steuern von iTunes und AppleTV mit Zugriff auf AirTunes-Lautsprechern von eben diesen Geräten. Wir zeigen Ihnen in diesem Kapitel, wie sie als Anwender von nomos diese Steuerung optimieren und entsprechende Vorteile sinnvoll nutzen und umsetzen können.

Da nomos nicht direkt auf die Remote Anwendung zugreifen kann, machen wir uns die Leistungsmerkmale des Eventservers zu nutze. Hierüber wird der `PLAYERSTATUS` der iTunes Anwendung überwacht. Sobald iTunes über Remote, lokal oder über Drittanwendungen angespielt wird, meldet nomos den Status über den Broadcast Port.

<ITUNES>GETPLAYERSTATE=PLAYING|OK</ITUNES> Diese Meldung kann mittels folgender `EventServer Definition` ausgewertet werden:

[CONFIG];

TRIGGERIP;192.168.50.55 lokale IP Adresse

TRIGGERPORT;1038 lokaler Broadcast Port

TRIGGERMODE;ASCII empfängt einen ASCII Code

[TRIGGERS]; <ITUNES>GETPLAYERSTATE=*|;iTunesRemoteEvent.myh;ONCHANGE

Die Trigger Definition bedeutet im Einzelnen:

<ITUNES>GETPLAYERSTATE=*|; Der Eintrag „*“ parst den Wert einer eingehenden Zeichenkette zwischen

<ITUNES>GETPLAYERSTATE=“ und dem Abschlusszeichen „|“. Bei Eintreffen des o.g. Status String wäre die Bedingung erfüllt und der Match wäre „`PLAYING`“

iTunesRemoteEvent.myh Ist der Name des Scriptfiles, das ausgeführt wird, wenn die Bedingung erfüllt wurde. In diesem Fall erstmal völlig unabhängig von dem Inhalt des „gematchten“ Strings.

ONCHANGE Damit jedoch das Script nur ausgeführt wird, wenn sich neben der erfüllten Bedingung auch das Match geändert hat, wird der Parameter `ONCHANGE` verwendet. Wie in dem Kapitel zu dem `EventServer` beschrieben, kann der Match als Übergabeparameter `[ARG]` für den Scriptaufruf verwendet werden.

Bedeutung der Scriptfiles:

Wir haben aus diesem Grund das aufzurufende Script, wie folgt aufgebaut, welches Ihnen die Funktionalität verdeutlichen soll.

<SCRIPT><RUN=iTunesRemoteEvent_[ARG].myh></SCRIPT>

Das Script `iTunesRemoteEvent.myh` dient in diesem Fall dazu, ein weiteres Script mit Parameteraufruf anzutriggern. Da das Match mit dem Wert „`PLAYING`“ gefüllt ist, erfolgt der Aufruf des Scriptfiles, sofern vorhanden, `iTunesRemoteEvent_PLAYING.myh`

Das Script `iTunesRemoteEvent_PLAYING.myh` beinhaltet somit die eigentliche Kommando- sequenz, die ausgeführt werden soll, wenn der iTUNES Status auf `PLAYING` gewechselt hat.

<ITUNES><FRONT><SHOW><MUTEOFF></ITUNES><SYS><OPENBUTTONS=ITUNES></SYS>
<BROWSER><HIDE></BROWSER><TV><MUTEON><HIDE></TV><HIDE><DVD><MUTEON>

```
<PAUSE><HIDE></DVD><SCRIPT><RUN=MRS_KUECHE(SC).myh><ARG=0></SCRIPT>
<SYS><DISABLESS><MUTEOFF></SYS>
```

... als Beispiel. Selbstverständlich kann für jede Statusausgabe ein entsprechendes SCRIPT mit entsprechenden Funktionen angelegt werden.

7.3 Nutzung der Squeeze Box als IR Empfänger zur Steuerung von nomos

Die Logitech Squeezebox verfügt über einen Infrarot-Empfänger. Dieser kann entsprechend ausgewertet werden um z.B. mit der Squeezebox Fernbedienung oder auch der kleinen Apple-IR-Remote das nomos System oder andere Geräte fernzusteuern!

Die Anwendung wird wie folgt eingerichtet:

Im Squeezecenter unter Einstellungen -> Plugins -> xPL-Schnittstelle die "Infrarot-Verarbeitung" auf "Raw" stellen und abspeichern. Die `xPL.csv` für die entsprechende Squeezebox um folgende Zeilen erweitern:

[SCHEMA=osd.basic];
OSDWRITE;COMMAND=write
OSDCLEAR;COMMAND=clear
OSDEXCLUSIVE;COMMAND=exclusive
OSDRELEASE;COMMAND=release
OSDTEXT;TEXT=\#
OSDROW;ROW=\#
OSDCOLUMN;COLUMN=\#
OSDDELAY;DELAY=\#
[WATCH=remote.basic];
keys;REMOTE_KEY

Eine `.csv` für den Eventserver anlegen, die die Infrarot-Kommandos in nomos-Befehle umsetzt (hier für die Apple-Remote):

<table>
<tr><th>[CONFIG];</th></tr>
<tr><td>TRIGGERIP;INTERNAL</td></tr>
<tr><td>TRIGGERPORT;BROADCAST</td></tr>
<tr><td>TRIGGERMODE;ASCII</td></tr>
<tr><td>[TRIGGERS];</td></tr>
<tr><td>REMOTE_KEY=77e150bf;<SYS><VOLUP=5></SYS></td></tr>
<tr><td>REMOTE_KEY=77e130bf;<SYS><VOLDN=5></SYS></td></tr>
<tr><td>REMOTE_KEY=77e160bf;<TV><CHUP></TV></td></tr>
<tr><td>REMOTE_KEY=77e190bf;<TV><CHDN></TV></td></tr>
<tr><td>REMOTE_KEY=77e1c0bf;<TV><FSOFF><FRONT><HIDE><MUTEON></TV></td></tr>
<tr><td>REMOTE_KEY=77e1a0bf;<TV><ON><SHOW><FSON><MUTEOFF></TV></td></tr>
<tr><td>GETCHNAME=*|;<SLIMSERVER><OSDTEXT=\n\#><OSDDELAY=10></SLIMSERVER></td></tr>
<tr><td>|VOL=*|;<SLIMSERVER><OSDTEXT=\nVolume:\#><OSDDELAY=10></SLIMSERVER></td></tr>
</table>

Die Klasse `<SLIMSERVER>` muß natürlich ggfs. angepasst werden, je nachdem wie die Squeezebox Definition benannt wurde.

Nach Neustart von nomos, die Apple-Remote auf die Squeezebox richten und EyeTV per Netzwerk über Infrarot fernbedienen:

Links/Rechts	Programme umschalten
Hoch/Runter	lauter/leiser
Play	TV an
Menu	TV aus

Als Feedback wird auf der Squeezebox der Sender bzw. die Lautstärke für 10 Sekunden angezeigt - eigentlich ziemlich sinnfrei, aber um zu zeigen, daß das bidirektional geht, durchaus brauchbar.

7.4 Nützliche Shortcuts und Tastaturbefehle

Es ist immer wieder beeindruckend wie viele versteckte und unbekannte Tastenkürzel, Tastenkombinationen, Shortcuts es unter Apple OS X bzw. dem neuen SnowLeopard gibt um Anwendungen unabhängig vom Einsatz der Maus oder des Touchpads zu steuern. Im Folgenden entsteht eine Auflistung aller möglichen Shortcuts, Tastaturkombinationen unter OS X/Snow Leopard und der darunter installierten Software um die tägliche Arbeitsweise an Ihrem Apple Computer noch effizienter gestalten zu können. (Quelle: `iFAQs.de`)

iFAQs.de - Shortcuts für den Systemstart von OS X/Snow Lepard:

[ALT]+[CMD]+O+F	Startet die Open Firmware Konsole beimSystemstart.
C	Der Rechner versucht zuerst vom CD-ROM Laufwerk zu booten
D	Der Rechner versucht zuerst von der internen Festplatte zu booten.
[ALT]+[CMD]+P+R	Löscht das Paramter RAM.
[SHIFT]+[ALT]+[CMD]	Backspace umgeht das aktuelle Startvolumen.
[ALT]	Öffnet bei neuem Rechner den Dialog zur Auswahl eines Startvolumens
[ALT]	Startvolumen bei Systemstart auswählen
[CMD]+V	Startet den Verbose-Modus
linke Maustaste	CD beim Systemstart auswerfen
T	Rechner im Targetmodus starten
[SHIFT]+[ALT]+[CMD]+[Del]	Umgeht das aktuelle Startvolumen und sucht nach einem alternativen Startvolumen.
N	Automatische Suche nach einem bootfähigen Netzlaufwerk
[SHIFT]	Systemstart im Safe Boot Modus
[CMD]+S	Startet den Single User Modus
D	Startet von der ersten Festplatten Partition
X	Startet OS X vom aktiven Startvolumen

iFAQs.de - Shortcuts für das Erstellen von Screenshots:

[SHIFT]+[CMD]+3	Screenshot des gesamten Bildschirms
[SHIFT]+[CTRL]+[CMD]+3	Screenshot wird automatisch in die Zwischenablage gesichert.
[SHIFT]+[CMD]+4	Auswahl mit Fadenkreuz welcher Bereich fotografiert werden soll.
[SHIFT]+[CMD]+4+Leertaste	Die Kamera fotografiert den aktuellen Bereich
[SHIFT]+[CMD]+F4	Das aktuelle Fenster wird fotografiert

7.5 Sonderzeichen (Mac/Win/VM-Ware)

Für das nomos Protokoll werden teilweise Sonderzeichen benötigt, die nicht auf der Tastatur vorhanden sind. Ebenso ist die Tastaturbelegung bei Benutzung einer virtuellen Maschine nicht konform der Apple Tastatur.

Nachfolgend zeigen wir die wichtigsten Tastaturbelegungen auf:

Zeichen	OS X	WIN
@	[ALT]+L	[ALT]+Q
/	[SHIFT]+7	[SHIFT]+7
\	[ALT]+[SHIFT]+7	[ALT]+ß
[	[ALT]+5	[ALT]+8
]	[ALT]+6	[ALT]+9
|	[ALT]+7	[ALT]+<
{	[ALT]+8	[ALT]+7
}	[ALT]+9	[ALT]+0

KAPITEL 8

nomos Cloud API

Mit der nomos Cloud API haben Entwickler oder externe Dienste Zugriff auf nomos gesteuerte Systeme.

Prinzipiell bietet der nomos Cloud API Dienst zwei unterschiedliche Schnittstellen. Einen klassischen REST Service und eine „REALTIME" Variante auf socket.io Basis. Der REST Service ist für die meisten Anforderungen ausreichend und eignet sich am Besten für die einfache Auslösungen/Ausführungen von Befehlen. Es können damit aber auch Eigenschaften und/oder aktuelle Zustände abgefragt werden.

Anmeldung unter: https://cloudapi.nomos-system.com/apirequest

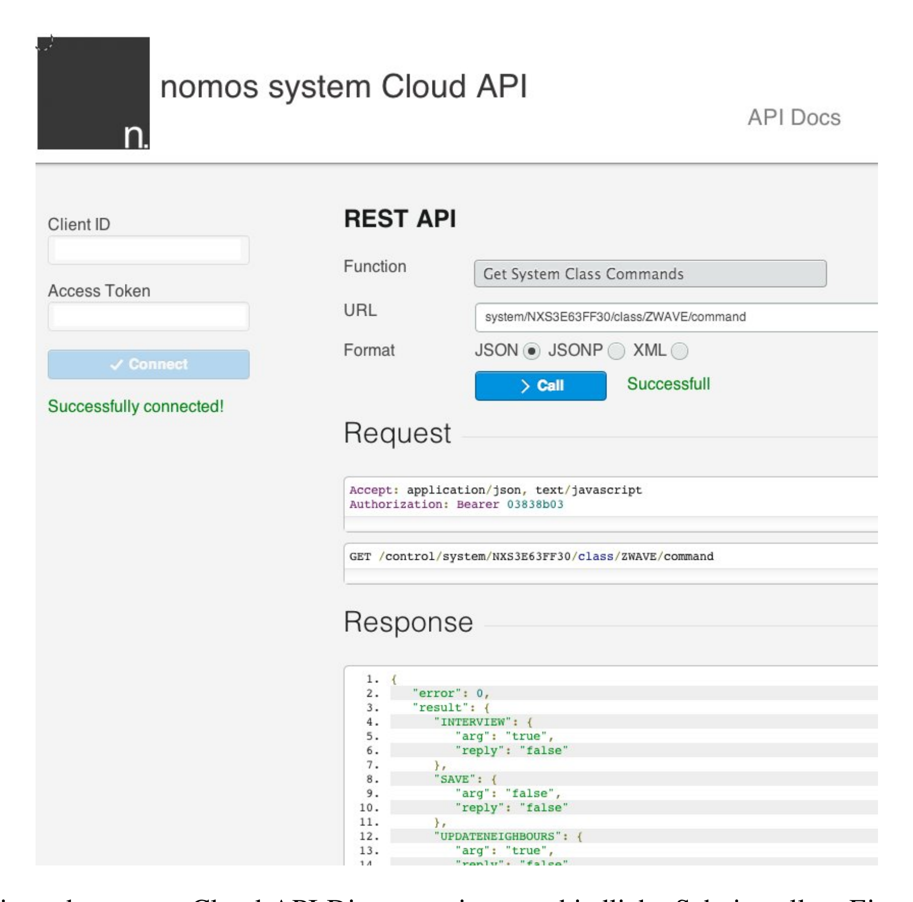
Mindestanforderung:

- Daemon 1.1.74
- HTML5 3.0.50

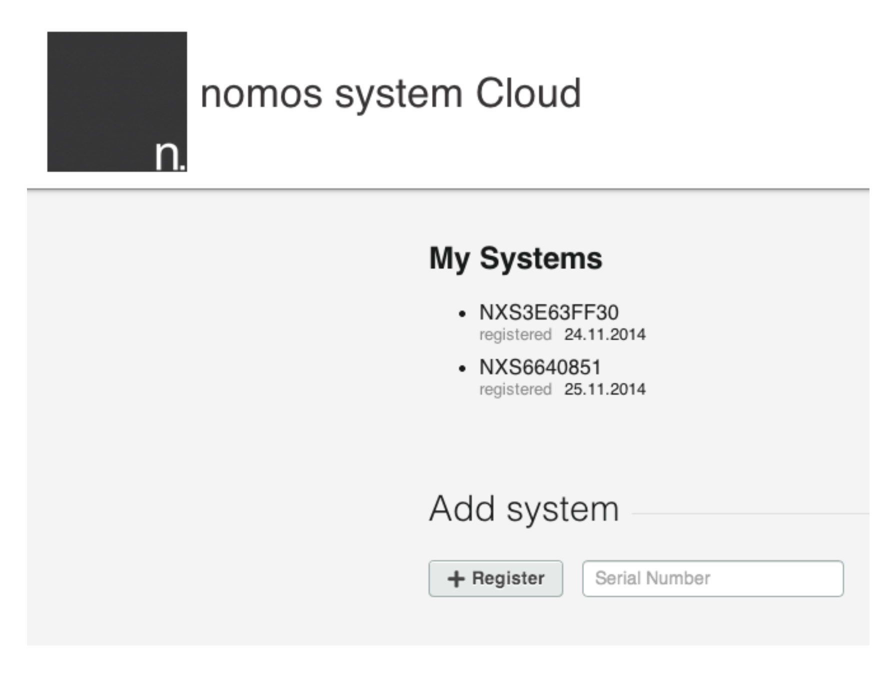
8.1 API Doku Allgemein

Mit der nomos Cloud API haben Entwickler oder externe Dienste Zugriff auf nomos gesteuerte Systeme.

Per OAuth 2.0 Authentifizierung ist sichergestellt, dass diese Clients nur auf berechtigte Benutzer und deren Systeme Zugriff haben.



Prinzipiell bietet der nomos Cloud API Dienst zwei unterschiedliche Schnittstellen. Einen klassischen REST Service und eine „REALTIME" Variante auf socket.io Basis. Der REST Service ist für die meisten Anforderungen ausreichend und eignet sich am Besten für die einfache Auslösungen/Ausführungen von Befehlen. Es können damit aber auch Eigenschaften und/oder aktuelle Zustände abgefragt werden.



Für zeitkritische Anforderungen oder in Fällen bei denen es nötig ist auf bestimmte Wert/Zustandsänderungen reagieren zu müssen ist das „REALTIME" Interface Ihre Wahl. Nähere Beschreibungen bzw. Details folgen in den weiteren Abschnitten.

OAuth 2.0 Authentifizierung

Das nomos Cloud API nutzt OAuth 2.0 als Basis zur Authentifizierung. Externe Applikationen können damit Zugriff auf Benutzer Accounts und Systeme beantragen ohne die Login Zugangsdaten zu kennen.

Um das OAuth Interface nutzen zu können wird eine „client ID" und „client secret" benötigt. Diese könnten unter folgender URL beantragt werden: https://cloudapi.nomos-system.com/apirequest

Standard Ablauf

Sie müssen folgende Schritte ausführen um eine API Anfrage stellen zu können:

1. Anfrage für den Zugriff auf einen nomos Account
2. nomos leitet den User auf die „Authorization" Seite weiter
3. Anfrage für einen Access Token
4. Empfang des Access Tokens
5. API Anfrage, siehe REST oder REALTIME API Abschnitte

1. Anfrage für den Zugriff auf einen nomos Account

GET https://cloudapi.nomos-system.com/oauth/authorise

Parameter

client_id Pflicht - Die „client ID" welche Sie von nomos erhalten haben

redirect_uri Pflicht - Der User wird nach Bestätigung auf diese URL weitergeleitet

Der Benutzer muss sich in diesem Schritt anmelden.

2. nomos leitet den User auf die „Authorization" Seite weiter

Wenn der Benutzer die Authentifizierung Anfrage bestätigt hat wird dieser auf die vorher definierte „redirect_uri" weitergeleitet. nomos fügt dieser URL den GET Parameter „code" hinzu. Dieser Code ist nur für wenige Sekunden gültig und dient zur Anfrage für einen Access Token. (siehe Schritt 3)

3. Anfrage für einen Access Token

Mit dem Code den Sie aus Schritt 2 erhalten haben können Sie einen Access Token anfragen.

POST https://cloudapi.nomos-system.com/oauth/token

Parameter

client_id Pflicht - Die „client ID" welche Sie von nomos erhalten haben

redirect_uri Pflicht - Die selbe URL wie in Schritt 1

client_secret Pflicht - Das „client secret" welches Sie von nomos erhalten haben

code Pflicht - Code den Sie in Schritt 2 erhalten haben

grant_type=authorization_code Pflicht

4. Empfang des Access Tokens

Sie erhalten folgende JSON Rückgabe wenn Sie erfolgreich einen Access Token angefragt haben:

```
{
        „token_type": "bearer",
        „access_token": "fd4889d3d6d329b67d209ec5834d7e04a17a8ab0",
        „expires_in": 3600,
```

```
        „refresh_token": "a0c88b520633d5ec5a39812da13377fb26426eb0"
}
```

Der Access Token ist nur für eine bestimmte Zeit gültig. Die Gültigkeitsdauer wird in Sekunden angegeben. In obigen Beispiel 3600 Sekunden, welches einer Stunde entspricht. Der „refresh_token" ist 2 Wochen gültig und kann verwendet werden um einen neuen Access Token anzufordern.

Access Token mit Hilfe des Refresh Tokens anfordern

Wenn der Access Token abgelaufen ist kann mit Hilfe des Refresh Token ein neuer Token angefragt werden.

POST https://cloudapi.nomos-system.com/oauth/refresh_token

Parameter

client_id Pflicht - Die „client ID" welche Sie von nomos erhalten haben

client_secret Pflicht - Das „client secret" welches Sie von nomos erhalten haben

refresh_token Pflicht - Der Refresh Token den Sie in Schritt 4 erhalten haben

Antwort Sie erhalten die selbe Antwort wie in Schritt 4, mit dem Unterschied das sich „access_token" und „refresh_token" geändert haben. Bitte beachten Sie das der alte Refresh Token dann nicht mehr gültig ist.

```
{
        "access_token": "c0d8b09a3d6867d31cbba64a3d5574a69eda9238",
        "expires_in": 3600,
        "token_type": "bearer",
        "refresh_token": "461690a37d06d0b994b008805ac1df025bbe72d8"
}
```

REST API

Jede Anfrage hat folgenden Aufbau: https://cloudapi.nomos-system.com/control/ + resource

z.B. um die Benutzer Informationen abzufragen: GET https://cloudapi.nomos-system.com/control/user

Der Access Token den Sie in OAuth Schritt 4 erhalten haben muss im HTTP Header „Authorization" verwendet werden: | Authorization: Bearer %access_token%.

Zusätzlich müssen Sie einen „Accept" Header definieren:

Accept: application/json, text/javascript
Authorization: Bearer a0c88b520633d5ec5a39812da13377fb26426eb0

HTTP-Headers

Accept „application/json, text/javascript" oder „application/xml, text/xml"
Das Rückgabe Format dass Sie vom Service erwarten.

Authorization Bearer %access_token%
Access Token aus der OAuth Autorisierung.

HTTP-Verbs

GET API Daten abfragen

POST API Anfrage ausführen

HTTP-Codes

Code	Beschreibung
200	Anfrage OK
304	Die Ressource hat sich nicht geändert
401	Der Access Token ist ungültig
403	Berechtigung Fehler
404	Die Ressource konnte nicht gefunden werden
500	Unerwarteter Fehler
503	Server nicht erreichbar (Wartungsmodus).

Callbacks (JSON-P)

Jede Anfrage kann um den Parameter „?callback" erweitert werden. Callbacks werden meist genutzt um „cross domain" Probleme zu verhindern. Bitte beachten Sie dass der Antwort Typ dann immer „application/javascript" ist. Für mehr Informationen siehe https://en.wikipedia.org/wiki/JSONP

Hinweis

Spezielle Parameter Namen müssen URL codiert angegeben werden. Im folgenden Beispiel ein KNX Adressen Name mit „/„ und Leerzeichen:

> system/NXS12345678/class/knx/value/1%2F2%2F3-Test%20Address

Verfügbare API Funktionen

siehe Testsuite Abschnitt

REALTIME API

Im Unterschied zu REST wird hier die nomos Cloud API Verbindung per socket.io (http://socket.io) einmalig aufgebaut und bleibt aufrecht erhalten. Ab diesen Zeitpunkt stehen unterschiedliche Funktionen/Befehle zur Verfügung um Informationen abzufragen oder zu senden.

Die große Stärke liegt aber in der „Listener" Funktionalität. Beschreibung siehe weiter unten.

Wichtig: nur socket.io kompatibel, nicht möglich per nativen WebSocket zu benutzen!

Einmalige Verbindungsherstellung

Die socket.io URL ist: https://cloudapi.nomos-system.com/control

Query Parameter

client Pflicht - Die „client ID" welche Sie von nomos erhalten haben

access_token Pflicht - Der Access Token den Sie in OAuth Schritt 4 erhalten haben

Beispiel mit jQuery: var socket = io.connect('https://cloudapi.nomos-system.com/control', {query: $.param({access_token: "a0c88b520633d5ec5a39812da13377fb26426eb0 ", client: "%CLIENT_ID%"})});

Beispiel ohne jQuery: var socket = io.connect('https://cloudapi.nomos-system.com/control', {query: "access_token=a0c88b520633d5ec5a39812da13377fb26426eb0&client=%CLIENT_ID%"});

Anfrage

Beispiel zur Abfrage von Benutzer Informationen:

```
socket.emit("getUser", {}, function(data) {
        alert(data);
});
```

Listeners

Für die Listener Funktionalität muss einmalig folgendes implementiert werden:

```
socket.on('listenerEvent', function (e) {
        alert(e);
});
```

Mit der Funktion „registerListener" kann nun ein „Listener" registriert werden.

Beispiel um bei iTunes Titeländerungen benachrichtigt zu werden:

```
socket.emit("registerListener", {
        "sid": „%SYSTEMID%", "class": "itunes", "value": "title"},
        function(data) {}
});
```

Der Event Listener „listenerEvent" wird nun automatisch bei Änderungen des Titels aufgerufen und die Daten können dort weiterverarbeitet werden.

Des weiteren stehen die Funktionen „unregisterListener" und „getRegisteredListener" zur Verfügung.

Verfügbare API Funktionen

siehe Testsuite Abschnitt

Testsuite

Für ein besseres Verständnis steht Entwicklern die sogenannte „Testsuite" zur Verfügung.

Diese ist erreichbar unter: https://cloudapi.nomos-system.com/testsuite

Die Testsuite bietet nicht nur eine Übersicht der vorhandenen Befehle und Funktionen, diese können auch direkt gleich getestet werden.

Hinweise

Wenn man mehrere Kommandos gleichzeitig einer Klasse aufrufen will nützt man am Besten die „RAW" Befehle.

Downloads:

Library für PHP https://github.com/nomos-system/cloudapi-lib-php

Library für JavaScript https://github.com/nomos-system/cloudapi-lib-js

8.2 API Befehlssatz

Stand 25.11.2014

Beschreibung	REST	REALTIME/WEBSOCKET
ALLGEMEIN		
liefert eine Liste der Systeme	GET /system	getSystems()
liefert ein bestimmtes System	GET /system/SYSID	getSystem({sid: SYSID})
liefert den Online Status eines bestimmten Systems	GET /system/SYSID/online	getSystemOnline({sid: SYSID})
liefert die Metadaten eines bestimmten Systems	GET /system/SYSID/meta	getSystemMeta({sid: SYSID})
liefert User Account Details	GET /user	getUser()
KLASSEN UND DEVICES		
liefert eine Liste der Devices/Klassen	GET /system/SYSID/class	getClasses({sid: SYSID})
liefert alle aktuellen Werte einer Klasse	GET /system/SYSID/class/CLASS/value	getClassValues({sid: SYSID, class: CLASS})
liefert bestimmten Wert einer Klasse	GET /system/SYSID/class/CLASS/value/VALUE	getClassValue({sid: SYSID, class: CLASS, value: VALUE})
liefert alle verfügbaren regulären Befehle einer Klasse	GET /system/SYSID/class/CLASS/command	getClassCommands({sid: SYSID, class: CLASS})
liefert alle verfügbaren SET Befehle einer Klasse	GET /system/SYSID/class/CLASS/set	getClassSetProperties({sid: SYSID, class: CLASS})
liefert alle verfügbaren GET Befehle einer Klasse	GET /system/SYSID/class/CLASS/get	getClassGetProperties({sid: SYSID, class: CLASS})
Aufruf JSON RAW Befehl	POST /system/SYSID/raw	execRAW({sid: SYSID, raw: {"version":1,"method":"command","class": "CLASS","command":[{"name": "COMMANDNAME", "value":"VALUE"}]}})
Aufruf eines bestimmten regulären Klassen Befehls samt Parametern	POST /system/SYSID/class/CLASS/command/ COMMANDNAME (PARAMETERS)	execClassCommand({sid: SYSID, class: CLASS, command: COMMANDNAME, param: PARAMETERS})
Aufruf eines bestimmten SET Klassen Befehls samt Parametern	POST /system/SYSID/class/CLASS/set/PROPERTYNAME (PARAMETERS)	execClassSetCommand({sid: SYSID, class: CLASS, property: PROPERTYNAME, param: PARAMETERS})
Aufruf eines bestimmten GET Klassen Befehls samt Parametern	POST /system/SYSID/class/CLASS/get/PROPERTYNAME (PARAMETERS)	execClassGetCommand({sid: SYSID, class: CLASS, property: PROPERTYNAME, param: PARAMETERS})
REALTIME EXCLUSIVE BEFEHLE		
Realtime Listener registrieren	——	registerListener({sid: „SYSID“, class: „CLASS“, value: "VALUE"})
Realtime Listener deregistrieren	——	unregisterListener({sid: „SYSID“, class: „CLASS“, value: "VALUE"})
liefert Liste der registrierten Listener	——	getRegisteredListener()
„listenerEvent“ wird automatisch aufgerufen bei Wertänderungen	——	EVENT: listenerEvent(DATA)

KAPITEL 9

Rechtliche Hinweise

Das in dieser Software und diesem Handbuch enthaltene Programmmaterial ist mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Die nomos system AG übernimmt folglich keine Verantwortung und wird keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung der Software/Handbuches oder Teilen davon entsteht.

Das Werk einschließlich aller Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung der nomos system AG unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Auf den Zeitpunkt der Verfügbarkeit von angekündigter Funktionalitäten sowie angekündigter Erweiterungen besteht kein wie auch immer gearteter Anspruch, solange keine offizielle Freigabe der nomos system AG vorliegt.

KAPITEL 10

Versions Historie

coming soon, bitte besuchen Sie unser Forum unter http://forum.nomos-system.com/

Anhänge (Dateivorlagen)

In den vorherigen Kapiteln wurden die Konfigurationsdateien im Detail erklärt. Nachfolgend haben wir Kopiervorlagen der verschiedenen Dateien mit allen verfügbaren Parametern angehangen. Zusätzlich sind hilfreiche Kommentartexte enthalten.

11.1 {commandserver}.csv

Dateiaufbau und mögliche Parameter der CommandServer Definitionen. Der Name der Datei definiert den Klassennamen.

Ort: ./[Projekt]/addons/

// Systemname, Hersteller, TYP

// Schnittstellen Parameter

// Ersteller

[CONFIG]	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
SERVERIP;192.168.1.200;	// entfernte {IP} oder {"LOCAL"}
SERVERPORT;6000;	// entfernter Port oder serielle Schnittstelle
SERVERPROTOCOL;UDP;	// {"UDP"}/{"TCP"}/{"HTTP"}/{"HTTPS"}/{"AUTO"}/{"SERIAL"}
SERVERMODE;ASCII;	// {"ASCII"}/{"HEX"}
SERVERTIMEOUT_IN_MS;1000;	// Timeout {0..3600} ms oder {"NONE"}

--- Erweitert (Optional)---

// MATCHING;	// Matching-Mode: {"FULL"}/{"NORMAL"}
// EXPORT;	// Freigabe des CommandServers (sharing option)
// CLIENTIP;	// weitere Client-IP, an die Replies gesendet werden
// CLIENTPORT;	// Port des Clients
// LOCALPORT;	// Lokaler Socket Port
// SERVERTIMEOUT;	// Wie TIMEOUT_IN_MS jedoch {0...3600} Sek. oder {"NONE"}
// SERVERTYPE;	// Checksummen Berechnung
// STARTUPCMD;	// Kommando bei Start bzw. autom. Verbindungsaufbau
// HEARTBEATCMD;	// {HeartbeatCMD};{Heartbeatinterval} in S. Default = 60s
// RECONNECT;	// {"YES" - Default/"NO"} auto. Herstellen der Verbindung nach Abbruch.

//	
// CMDPREFIX;	// Prefix fuer jedes Kommando
// CMDSUFFIX;	// Suffix fuer jedes Kommando
// SEQUENCEPREFIX;	// Prefix fuer jede Kommandosequenz
// SEQUENCESUFFIX;	// Suffix fuer jede Kommandosequenz
//	
// STARTDELIMITER;	// Begrenzer/ Trennzeichen am Anfang der Match Sequenz
// ENDDELIMITER;	// Begrenzer/ Trennzeichen am Ende der Match Sequenz
// MAPPREFIX;	// Prefix fuer jedes Match
// MAPSUFFIX;	// Suffix fuer jedes Match
//	
// SPACING;	// Kommando Verzögerung (Sequenzen) {t in ms}
//	
// USERNAME;	// Sofern Authentifizierung notwendig - Username
// PASSWORD;	// Sofern Authentifizierung notwendig - Passwort

[COMMANDS];

// Aufbau:

// {mmh-Command};{Protokollsequenz};{Different Command Timeout};{Parameter}

[MAPPINGS];

// Aufbau:

// {Protokoll-Match};{mmh-Bezeichnung};{local Action};ONCHANGE;{Match-Bedingung}

11.2 {buttons}.csv (Mac only)

Dateiaufbau und mögliche Parameter der Button Definitionen (Mac only). Der Name der Datei definiert den Namen der Buttonleiste.

Ort: ./[Projekt]/gui/

// Ersteller

[CONFIG]	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
TIMEOUT;5;	// Zeit bis zum autom. Schließen der Leiste
POSITION;BUTTOM;	// Position der Leiste {LEFT/RIGHT/TOP/BOTTOM (default)}
SIZE;AUTO;	// Hoehe der Leiste in Pixeln AUTO=default
CLOSEBUTTON;YES;	// Leiste schließbar (nur bei POSITION;BUTTOM)
HUD;YES;	// HUD Modus (nur bei POSITION;BUTTOM)
TRANS;0.95;	// Transparenz der Buttonleiste {0.1...1.0}(default: 0.8)
RED;0.16;	// Rot-Anteil der Buttonleiste
GREEN;0.16;	// Gruen-Anteil der Buttonleiste
BLUE;0.16;	// Blau-Anteil der Buttonleiste
TEXTRED;0.16;	// Rot-Anteil der Buttontext
TEXTGREEN;0.16;	// Gruen-Anteil der Buttontext
TEXTBLUE;0.16;	// Blau-Anteil der Buttontext
// TEXTBACKGROUNDRED;	// Rot-Anteil der Buttontext Hintergrund
// TEXTBACKGROUNDGREEN;	// Gruen-Anteil der Buttontext Hintergrund
// TEXTBACKGROUNDBLUE;	// Blau-Anteil der Buttontext Hintergrund
// TEXTSIZE;	// Zeichengröße des Buttontext (default: dynamisch)
IMAGE;SCALE;	// {SCALE (default) /CENTER/TOP/BOTTOM/LEFT/RIGHT}

[BUTTONS];

// Belegung:

// {Label1};{Action1};{Buttontyp};{Action2};{Grafik1};{Label2};{Grafik2}

11.3 webserver.csv

Dateiaufbau und mögliche Parameter der Webserver Definitionen. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// mini Webserver

//

// Hinweis: TV Senderlogos muessen unter ../misc/web/tv_{Kanalname}.jpg(png) abgelegt werden.

// {Kanalname} ist der identische Name des Senders aus eyeTV.

//‘

// eyeTV URL’s:

// http://{IP:PORT}/tv_current.jpg(png)

// http://{IP:PORT}/tv_{channel Name}.jpg(png)

//

// iTunes URL’s:

// http://{IP:PORT}/itunes_current.jpg(png)

// http://{IP:PORT}/iTunes_ID_{ID}.jpg(png)

// http://{IP:PORT}/iTunes_album_{Ablum Name}.jpg(png)

//

// AppleTV (remote) URL’s:

// http://{IP:PORT}/[ATV CLASS]_current.jpg(png)

//

// SONOS URL’s:

// http://{IP:PORT}/[SONOS CLASS]_current.jpg(png)

//

// DVD Player URL’s:

// benoetigt "preview.jpg" im DVD Ordner

// http://{IP:PORT}/dvd_current.jpg(png)

// http://{IP:PORT}/dvd_{dvd title}.jpg(png)

//

// Genereller Hinweis:

// - URL is not case senstive

// - dummy file support

// ../misc/web/tv_dummy.jpg(png) - fuer eyeTV

// ../misc/web/itunes_dummy.jpg(png) - fuer iTunes

// ../misc/web/dvd_dummy.jpg(png) - fuer DVD Player

//

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {“YES” - Default/”NO”}
DEBUG;NO;	// Erweitertes Protokoll aktiv {“YES”/”NO” - Default}
PORT;1234;	// Webserver Port
GUIPORT;1235;	// GUI download Port oder PORT +1
// DVD_COVER;	// Legt den DVD Covernamen fest (Default: preview.jpg)
// DVD_PATH;	// Definition bei lokaler URL {Users/[PROFILE]/Movies/}

[ALIASES];

//{Aliasname};{URL};

11.4 sysvars.csv

Dateiaufbau der sysvar.csv. Der Dateiname ist zwingend vorgegeben. Der Inhalt der Variablen kann zur Laufzeit verändert werden.

Ort: ./[Projekt]/misc/

// Systemseitig reservierte und automatisch erzeugte Variablen:

// [SYSTEM_IP]	- aktuelle IP
// [SCREEN_X]	- aktuelle x-Auflösung des Bildschirms
// [SCREEN_Y]	- aktuelle y-Auflösung des Bildschirms
// [BUTTON_SIZE]	- aktuelle Höhe der Buttonleiste
// [HOMEDIR]	- aktuelles Verzeichnis
// [PROFILE]	- aktuelle Profilname (Kurzname)
// [PROJECT]	- aktueller Projektpfad
// [TIME]	- aktuelle Zeit im KNX EIS3 Format
// [DATE]	- aktuelles Datum im KNX EIS3 Format
// [OEM_NAME]	- Name der OEM Version
// [SYSTEM_SERIAL]	- Seriennummer der aktuellen Hardware
// [SYSTEM_MAC]	- MAC Adresse der Netzwerkschnittstelle
//	
// [ARG]	- interne Zwischenablage Argumentpuffer
// [GARG]	- interne Zwischenablage Argumentpuffer (global)
//	

//**

// Profil spezifische Variablen

//**

// Systemvariablen {Name};{Wert}

SYSTEMID;Vorlage // Name des Systems (Anzeige im ScriptClient)

11.5 timer.csv

Dateiaufbau der timer.csv. Der Dateiname ist zwingend vorgegeben. Die Zeitprogramme und die Timer können zur Laufzeit verändert werden.

Ort: ./[Projekt]/misc/

// Timer:

// {Name};{Typ};{Wert o.Datums-String};{Action};{Optionen}

// {Name}	Bezeichnung des Timers
// {Typ}	Typ des Timers, “INTERVAL”,”ONESHOT”,”CALENDAR”
// {Wert}	Konfiguration des Timerwertes, bei “INTERVAL” oder “ONESHOT”:
//	Sekunden (max. 86400 (= 1Tag) möglich)
// {Datums-String}	Konfiguration des Timerwertes, bei “CALENDAR”:
//	{nichts} - feuert jeden Tag um 0 Uhr
//	13 - feuert jeden Tag um 13 Uhr
//	13:04 - feuert jeden Tag um 13 Uhr und 4 Minuten
//	13:04:05 - feuert jeden Tag um 13 Uhr, 4 Minuten und 5 Sekunden
//	MON - feuert jeden Montag um 0 Uhr
//	WED,13:04 - feuert jeden Mittwoch um 13:04 Uhr
//	WED,MON,13 - feuert jeden Mittwoch und Montag um 13 Uhr
//	THU,13,29.11. - feuert jeden Donnerstag um 13 Uhr,
//	da Wochentag angegeben (Vorrang)
//	13,12. - feuert jeden 12. des Monats um 13 Uhr
//	24.12.,18 - feuert jeden Heiligabend um 18:00 Uhr
//	13,25.12. - feuert am 1. Weihnachtsfeiertag um 13:00 Uhr
//	
// Syntaxvarianten:	{Datums-String}~{Varianz}
//	
//	~{Sekunden}
//	~{Minuten}:{Sekunden}
//	~{Stunden}:{Minuten}:{Sekunden}
//	
// {Action}	auszuführende mmh-Sequenz bzw. Scriptname
// {Optionen}	weitere Optionen zur Timerkonfiguration, z.Zt. möglich.
//	Mehrfachauswahl werden durch ”,” getrennt:
//	AUTOSTART - startet den Timer beim Systemstart
//	IMMEDIATE - feuert den Timer auch bei Systemstart
//	
[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {“YES” - Default/”NO”}
DEBUG;NO;	// Erweitertes Protokoll aktiv {“YES”/”NO” - Default}

[TIMERS];

// {Name};{Typ};{Wert};{Action};{Optionen}

// Beispiele:

// TIME;INTERVAL;60;<SYS><GETTIME><GETDATE></SYS>;AUTOSTART;

// permanent Timer, der alle 60s die Systemzeit abfragt

// CAL;CALENDAR;21:35;<SYS><SAY=Hello></SYS>;AUTOSTART;

    // Feuert täglich um 21:35

// CAL;CALENDAR;22:30~600;<SYS><SAY=Hello></SYS>;AUTOSTART;

    // Timer mit Varianz Feuert täglich zwischen 22:25 und 22:35

11.6 baos.csv

Dateiaufbau der baos.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung des Weinzierl BAOS 770 Interface (Objektserver)

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
BAOSIP;AUTO	// {IP} - IP des KNXBAOS (fest)
-	// {AUTO} - IP des BAOS wird automatisch gesucht
-	// {OFF}- BAOS-Funktion abgeschaltet

[DATAPOINTS];

// Angemeldete Datenpunkte, wie in der ETS konfiguriert

// {Objektnummer};{frei wählbarer Datenpunkt-Name};{Object Typ "DEC","HEX", "ASCII"}

[ACTIONS];

// {Objektnummer};{Wert};{Aktion};{Object Typ "DEC","HEX", "ASCII"}

// Wenn {Object Typ} nicht gesetzt gilt default = HEX

// Beispiele:

// 1;1;<SCRIPT><RUN=MainButton_iTunes.myh></SCRIPT>;DEC;

    // Startet ein Script

// 2;#;<SYS><VOLSET=\#></SYS>;DEC;

    // Setzt die Systemlautstärke auf den Wert von 5/2/2

11.7 hs.csv

Dateiaufbau der hs.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung des Gira Homesever KO-Gateway's

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
HSIP;192.168.50.65;	// IP Adresse GIRA Home Servers
HSPORT;7003;	// Port des KO-Gateway's
HSKEY;	// Key, wenn vergeben
ADDRTYPE;3STEP	//{2STEP} 2-stufig, {3STEP} 3-stufig, {leer} dezimal/raw

[DATAPOINTS];

// {internes oder externes Objekt};{frei wählbarer DP Name};{Bedingung};{Aktion}

//Beispiele

// 5/2/1;iTunes Start;1.0;<SCRIPT><RUN=MainButton_iTunes.myh></SCRIPT>;

// Startet ein Script, wenn 5/2/1 = "1"

// 5/2/2;Set Volume;;#;<SYS><VOLSET=\#></SYS>;

// Setzt die Systemlautstärke auf den Wert von 5/2/2

// 5/2/8;Volume down;DOWN,100;<SYS><VOLDN=5></SYS>;

// Ändert die Systemlautstärke rel. bei einem Dim- Befehl

// 5/2/8;Volume up;UP,100;<SYS><VOLUP=5></SYS>;

// Ändert die Systemlautstärke rel. bei einem Dim+ Befehl

11.8 knx.csv

Dateiaufbau der knx.csv. Der Dateiname ist zwingend vorgegeben. Es wird zus. die .esf Datei benötigt, die aus der ETS generiert werden kann. Die .esf Datei muss im selben Verzeichnis liegen, wie die knx.csv.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung des KNXnet/IP Protokolls

// Unterstützte EIS Typen in der esf Datei:

//

// EIS 1 'Switching' (1 Bit)

// EIS 2 'Dimming - position' (1 Bit)

// EIS 2 'Dimming - control' (4 Bit)

// EIS 2 'Dimming - value' (8 Bit)

// EIS 3 'Time' (3 Byte)

// EIS 4 'Date' (3 Byte)

// EIS 5 'Value' (2 Byte)

// EIS 6 'Scaling - percent' (8 Bit)

// EIS 6 'Scaling - degree' (8 Bit)

// EIS 7 'Drive control' (1 Bit)

// EIS 8 'Priority - position' (1 Bit)

// EIS 8 'Priority - control' (2 Bit)

// EIS 9 'Float value' (4 Byte)

// EIS 10 '16Bit Counter' (2 Byte)

// EIS 11 '32Bit Counter' (4 Byte)

// EIS 12 'Access' (4 Byte)

// EIS 13 'EIB-ASCII-Char' (8 Bit)

// EIS 14 '8Bit Counter' (8 Bit)

// EIS 15 'Character String' (14 Byte)

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
ESF;knx.esf;	// Name der esf Datei
PHYSADDR;0.0.254;	// Physikalische. Adresse des Daemon
ADDRTYPE;3STEP;	// {3STEP} o. {2STEP} Darstellung der Gruppenadressen
MATCHLOCAL;NO;	// Match auch auf Daten, die vom Daemon gesendet werden
DPTNAMES;NO	// Ausgabe mit Bezeichnungstext
SENDRATE;10	// {30} = Default, (Telegramme/s)
INITSCAN;YES;	// scannt die definierten Tabellen beim Engine-Start // „YES/ALL" = ein, „NO" = aus (Default)

// ****Einstellung für ROUTING ****

CONNECTIONTYPE;ROUTING	// {„TUNNELING"/„ROUTING"-Default} o. „TUNNELING_BRIDGE"
MULTICAST_IP;224.0.23.12	// {Multicast Adresse} oder {„AUTO"– default}

```
// **** Einstellung für TUNNELING ****
//CONNECTIONTYPE;TUNNELING
//DEVICE_IP;AUTO
// **** Einstellung für TUNNELING_BRIDGE ****
//CONNECTIONTYPE;TUNNELING_BRIDGE;
//DEVICE_IP;AUTO;
//MULTICAST_IP;AUTO;
[DATAPOINTS];
// {KNX/EIB Gruppenadresse};{Bedingung};{mmh-Sequenz oder Scriptname};
// Beispiele:
// 5/2/1;1;<SCRIPT><RUN=script.myh></SCRIPT>;
        // Startet ein Script, wenn 5/2/1 = "1"
// 5/2/1;0; <SCRIPT><RUN=script.myh></SCRIPT>;
        // Startet ein Script, wenn 5/2/1 = "0"
// 5/2/2;#;<SYS><VOLSET=\#></SYS>;
        // Setzt VOLSET auf den Wert von 5/2/2
// 5/2/8;DOWN,100;<SYS><VOLDN=5></SYS>
        // VOLDN bei rel. DIM Befehl
// 5/2/8;UP,100;<SYS><VOLUP=5></SYS>;
        // VOLUP bei rel. DIM Befehl
// [TABLE=Wohnzimmer]; // 1/8/4
        // Data Scan Adress Tabelle
```

11.9 remote.csv

Dateiaufbau der remote.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung der remote Klassen.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}

[DEVICES];

// {Klassenname};{IP Adresse AppleTV oder entferntes iTunes};

//

// ATV_FIT;192.168.1.40;

11.10 xbmc.csv

Dateiaufbau der xbmc.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung der xbmc Klassen.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}

[DEVICES];

// {Klassenname};{IP};{ControlPort};{WebserverPort};{Username};{Passwort};

// {IP}	IP des XBMC
// {ControlPort}	Steuer Port des XBMC (optional, default: 9090)
// {WebserverPort}	Port des Webservers (optional, default: 8080), ArtInfo
// {Username}	Username für den Webserver (optional, default "xbmc")
// {Passwort}	Passwort für den Webserver (optional, default "xbmc")
//	
// XBMC_FIT;192.168.1.40;	

11.11 sonos.csv

Dateiaufbau der sonos.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung der remote Klassen.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {“YES” - Default/”NO”}
DEBUG;NO;	// Erweitertes Protokoll aktiv {“YES”/”NO” - Default}

[DEVICES];

// {Klassenname};{IP Adresse oder Name/Bezeichnung des SONOS Player};

//

// SON_FIT;Fitness;

11.12 mremote.csv

Dateiaufbau der mremote.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung des mremote Moduls.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
PORT;5500;	// Protokoll Port
//PORTRAIT;;	// {Local-Action bei iPxx Orientierung hochkant
//LANDSCAPE;;	// {Local-Action bei iPxx Orientierung quer
//CONNECT;;	// {Local-Action, bei iPxx Verbindung
//SPEECHLANGUAGE; {Lang-Setting}	// waehlt die Sprache fuer die Sprachausgabe (default: en_US)
//SPEECHVOICE;{Voice}	//wählt die Stimme für die Sprachausgabe (default: (null))

// Eine Liste der möglichen Sprachen und Stimmen gibt es hier

GUI_IP;AUTO;	// IP Adresse des mRemote Servers, oder AUTO (AUTO=lokale IP Adresse)
GUI_PORT;AUTO;	// Port Adresse des mRemote Servers oder AUTO (AUTO=Port des lokalen mRemote Servers)

[DIGITAL];;

// {JoinNr};{Type};{Push-Action};{Release-Action};{Push-Event};{Release Event};NOCACHE;{Local-Action};{Parameter};

[ANALOG];;;

// {JoinNr};{value};{Scale};{Action};{Event};NOCACHE;{Local-Action};{Parameter};

[SERIAL];;;

// {JoinNr};{value};{Action};{Event};NOCACHE;{Local-Action};{Parameter};

[LIST];;;

// {JoinNr}};{value};{Action};{Event};NOCACHE;{Local-Action};{Parameter}

[PAGE];;;

// {Page Match};{Action};

11.13 logic.csv

Dateiaufbau der logic.csv. Der Dateiname ist zwingend vorgegeben.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung des Logik Moduls.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}

[OBJECTS];

// {Objektname};{Match};{Startwert};{Kanal};ONCHANGE;

// {Objektname} -	- Name des Objektes
// {Match}	- Match-String fuer den Wert des Objektes
// {Startwert}	- Vorbelegung des Objektwertes (optional, default: 0)
// {Kanal}	- Parser-Kanaele (IN/OUT/BC), auf denen gematcht werden soll (optional, default: IN,OUT,BC)
// ONCHANGE	- triggert Logiken (siehe unten) nur bei Werteaenderungen

// Beispiel:

//MYVOL;<SYS>\%|VOL=*|;;BC,OUT;ONCHANGE;

// matcht auf die Systemlautstaerke im OUT- und BC-Kanal und Text hier eingeben

beschreibt Objekt MYVOL

[LOGIC];

// {logischer Ausdruck};{positive Aktion};{negative Aktion};{Triggerliste};ONCHANGE

// {logischer Ausdruck} -	Verknuepfungen von Objekten, Sysvars und Konstanten
// {positive Aktion}	- Aktion, die bei einem logischen "wahr" des {logischen Ausdrucks} ausgefuehrt werden soll, Objekte koennen mit [[ ]] eingefuegt werden.
// {negative Aktion}	- Aktion, die bei einem logischen "falsch" des {logischen Ausdrucks} ausgefuehrt werden soll, Objekte koennen mit [[ ]] eingefuegt werden
// {Triggerliste}	- Liste aller Objekte und Sysvars, die die Evaluierung des {logischen Ausdrucks} anstossen (optional, default: alle Objekte und Sysvars aus {logischer Ausdruck})
// ONCHANGE	- triggert die positive bzw. negative Action nur dann, wenn sich der Wert des logischen Ausdrucks geaendert hat (wahr -> falsch oder falsch -> wahr) (optional)

// Operanden von {logischer Ausdruck}:

<table>
<tr><th>// {Objektname}</th><th>- Inhalt des Objektes {Objektname} oder JoinNr {dxxx, axxx, sxxx}</th></tr>
<tr><td>// [{Sysvarname}]</td><td>- Inhalt der Sysvar {Sysvarname}</td></tr>
<tr><td>// "{Konstante}"</td><td>- konstanter Wert</td></tr>
<tr><td>//</td><td></td></tr>
<tr><td>// logische Verknuepfungen:</td><td></td></tr>
<tr><td>// ! - "NOT"</td><td>- logisches NICHT, kann vor Operanden und Klammern stehen</td></tr>
<tr><td>// | oder ||</td><td>- "ODER", Operand a oder Operand b sind wahr</td></tr>
<tr><td>// & oder &&</td><td>- "UND", Operand a und Operand b sind wahr</td></tr>
</table>

//	
// Vergleicher:	
// = oder == - “GLEICH”	- Operand a ist gleich Operand b
// != oder <> - “UNGLEICH”	- Operand a ist ungleich Operand b
// > oder >> - “GROESSER”	- Operand a ist groesser Operand b (nur numerisch)
// < oder << - “KLEINER”,	- Operand a ist kleiner Operand b (nur numerisch)
// >= oder => - “GROESSER-GLEICH”	- Operand a ist groesser o.gleich Operand b (nur numerisch)
// <= oder =< - “KLEINER-GLEICH”	- Operand a ist kleiner oder gleich Operand b (nur numerisch)
// Rechenvorschrift:	
// ()	- die Evaluierung von gesetzten Klammern hat Vorrang

11.14 zwave.csv

Dateiaufbau der zwave.csv. Der Dateiname ist zwingend vorgegeben. Im Konfigurationsmodus ist keine csv erforderlich. Im Konfigurationsmodus wird diese Datei autom. angelegt.

Ort: ./[Projekt]/misc/

// Definitionstabelle für die Verwendung der Z-Wave Klassen.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {"YES" - Default/"NO"}
DEBUG;NO;	// Erweitertes Protokoll aktiv {"YES"/"NO" - Default}
PORT;AUTO	// "AUTO" oder serieller Port, an dem der zwave Controller angeschlossen ist (/dev/ttyS1, /dev/ttyUSB0 usw...), optional, default: AUTO

[NODES];

// {NodeIndex};{gewünschter Nodename}

// 5;DIMMER;

// 6;SWITCH;

11.15 xpl.csv

Dateiaufbau, mögliche Parameter und Befehlssatz der xpl Definitionen. Der Name der Datei definiert den Klassennamen. Nachfolgendes Beispiel ist für die Squeezbox von Logitech

Ort: ./[Projekt]/xpl/

// Definitionstabelle für die Verwendung der Z-Wave Klassen.

[CONFIG];	
ACTIVE;YES;	// Modul aktivieren {“YES” - Default/”NO”}
DEBUG;NO;	// Erweitertes Protokoll aktiv {“YES”/”NO” - Default}
TARGET;slimdev-slimserv.SB1	// Name der Squeeze Box

[SCHEMA=audio.slimserv];
PLAY;COMMAND=PLAY;
STOP;COMMAND=STOP;
VOLUME;COMMAND=VOLUME \#;
NEXT;COMMAND=SKIP;
PREV;COMMAND=BACK;
RANDOM;COMMAND=RANDOM;
CLEAR;COMMAND=CLEAR;
PAUSETOGGLE;EXTENDED=pause;
PAUSEON;EXTENDED=pause 1;
PAUSEOFF;EXTENDED=pause 0;
PLAYFILE;EXTENDED=playfile \#;
SETPTIME;EXTENDED=time \#;
SLEEP;EXTENDED=sleep \#;
POWERON;EXTENDED=power 1;
POWEROFF;EXTENDED=power 0;
ADDANDPLAY;EXTENDED=playlist play \#;
ADD;EXTENDED=playlist add \#;
INSERT;EXTENDED=playlist insert \#;
MOVE;EXTENDED=playlist move \# \#;
DELETE;EXTENDED=playlist delete \#;
RESUME;EXTENDED=playlist resume \#;
SAVE;EXTENDED=playlist save \#;
LOADALBUM;EXTENDED=playlist loadalbum \# \# \#;
ADDALBUM;EXTENDED=playlist addalbum \# \# \#;
JUMP;EXTENDED=playlist index \#;
SHUFFLEON;EXTENDED=playlist shuffle 1;
SHUFFLEOFF;EXTENDED=playlist shuffle 0;
NOREPEAT;EXTENDED=playlist repeat 0;
REPEATSONG;EXTENDED=playlist repeat 1;
REPEATPLAYLIST;EXTENDED=playlist repeat 1;
RESCAN;EXTENDED=rescan;

[SCHEMA=audio.request];

STATUS;COMMAND=STATUS;

[WATCH=audio.basic];	// Verarbeitung der Ruckmeldungen
status;STATE;	
ARTIST;ARTIST;	
ALBUM;ALBUM;	
TRACK;TITLE;	
POWER;POWER;	